

NUMERICALLY APPROXIMATING ZEROS OF A POLYNOMIAL

By

RANDY JOE SNIDER

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1968

Submitted to the Faculty of the Graduate School
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
May, 1970

OKLAHOMA
STATE UNIVERSITY
LIBRARY
OCT 15 1970

NUMERICALLY APPROXIMATING ZEROS OF A POLYNOMIAL

Thesis Approved:

Craig A. Wood

Thesis Adviser

James D. Whitfield

Paul G. Miller

D. Dunbar

Dean of the Graduate College

762807

PREFACE

In this study I have presented five iterative methods for approximating zeros of a polynomial using the digital computer. Chapter I is an introduction to the material which follows. Chapters II through VI contain the algorithms and convergence properties for Newton's, Muller's, Greatest Common Divisor, Lehmer's and the Quotient-Difference methods, respectively. Chapter VII compares Newton's, Muller's, and the Greatest Common Divisor methods, giving their advantages, disadvantages, and results of computer tests performed. Appendices B through E contain flow diagrams, program listings, and instructions for use of the programs for Newton's, Muller's, Greatest Common Divisor method used with Newton's method, and Greatest Common Divisor method used with Muller's method, respectively. Appendix A describes a number of special features performed by the programs. Appendix F presents flow diagrams, program listings, and instructions for use of a program to construct the polynomial resulting from the product of a given number of linear factors.

I would like to express my appreciation to Dr. Craig A. Wood for his assistance and guidance in directing the writing of this thesis.

Also I would like to thank the National Aeronautics and Space Administration for the financial support received as graduate assistant on research grant NASA NGR 37-002-084; to Dr. Wood for the opportunity of working with him on his grant; and to Dr. John W. Jewett, Head of the Department of Mathematics, for my teaching assistantship.

Furthermore, I would like to thank Dr. Forrest D. Whitfield and Dr. Paul G. Miller for serving as chairman and member, respectively, of my advisory committee.

In addition, I would like to thank Beverly Turner, Mathematics Departmental Secretary, for an excellent job done in typing the manuscript.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION	1
II. NEWTON'S METHOD	5
Derivation of the Algorithm	5
Convergence of Newton's Method	9
Procedure for Newton's Method	10
Geometrical Interpretation of Newton's Method	11
Determining Multiple Roots	11
III. MULLER'S METHOD	13
Derivation of the Algorithm	13
Procedure for Muller's Method	20
Geometrical Interpretation of Muller's Method	21
Determining Multiple Roots	21
IV. GREATEST COMMON DIVISOR METHOD	23
Derivation of the Algorithm	23
Determining Multiplicities	34
Procedure for the G.C.D. Method	35
V. LEHMER'S METHOD	36
Derivation of the Algorithm	36
Convergence of Lehmer's Method	47
Determining Multiplicities	47
Procedure for Lehmer's Method	47
Conclusion of the Method	50
VI. QUOTIENT-DIFFERENCE METHOD	51
Derivation of the Algorithm	51
Existence of the Q-D Scheme	64
Convergence Properties	64
Zeros of Equal Magnitude	66
Zero Coefficients of the Polynomial $P(X)$	68
Procedure for the Q-D Method	71
Conclusion of the Method	74

Chapter	Page
VII. CONCLUSION	76
Polynomials With all Distinct Roots	76
Polynomials With Multiple Roots	78
Polynomials With Roots Close Together	83
BIBLIOGRAPHY	119
APPENDIX A - SPECIAL FEATURES OF THE PROGRAMS	120
Generating Approximations	120
Altering Approximations	122
Searching the Complex Plane	123
Improving Zeros Found	125
Solving Quadratic Polynomial	125
Missing Roots	126
Miscellaneous	126
APPENDIX B - NEWTON'S METHOD	134
Use of the Programs	134
Input Data for Newton's Method	138
Variables Used in Newton's Method	143
Description of Program Output	154
Informative and Error Messages	155
APPENDIX C - MULLER'S METHOD	179
Use of the Programs	179
Input Data for Muller's Method	184
Variables Used in Muller's Method	184
Description of Program Output	185
Informative and Error Messages	186
APPENDIX D - G.C.D. - NEWTON'S METHOD	219
Use of the Programs	219
Input Data for G.C.D. - Newton's Method	222
Variables Used in G.C.D. - Newton's Method	226
Description of Program Output	226
Informative and Error Messages	226
APPENDIX E - G.C.D. - MULLER'S METHOD	268
Use of the Programs	268
Input Data for G.C.D. - Muller's Method	270
Variables Used in G.C.D. - Muller's Method	271
Description of Program Output	271
Informative and Error Messages	271

Chapter	Page
APPENDIX F - POLYNOMIAL GENERATOR	312
Use of the Programs	312
Input Data for the Polynomial Generator	313
Variables Used in the Polynomial Generator	316
Description of Program Output	316

LIST OF TABLES

Table	Page
I. Program Changes for Solving Polynomials of Degree Greater Than 25 by Newton's Method	135
II. System Functions Used in Newton's Method	136
II-A. System Functions in Newton's Method to be Typed When the WATFOR Compiler is Used	137
III. System Functions for Converting Single Precision Newton's Method to Double Precision	138
IV. Control Data for Newton's Method	139
V. Coefficient Data for Newton's Method	141
VI. Initial Approximation Data for Newton's Method	142
VII. Data to End Execution of Newton's Method	143
VIII. Variables Used in Newton's Method	148
VIII-A. Single Precision Program for Newton's Method	163
VIII-B. Double Precision Program for Newton's Method	171
IX. Program Changes for Solving Polynomials of Degree Greater Than 25 by Muller's Method	180
X. System Functions Used in Muller's Method	181
X-A. System Functions in Muller's Method to be Typed When the WATFOR Compiler is Used	182
XI. System Functions for Converting Single Precision Muller's Method to Double Precision	183
XII. Correspondence of Newton's and Muller's Input Data Variables	185
XIII. Variables Used in Muller's Method	187
XIII-A. Single Precision Program for Muller's Method	197

Table	Page
XIII-B. Double Precision Program for Muller's Method	206
XIV. Program Changes for Solving Polynomials of Degree Greater Than 25 by G.C.D. - Newton's Method	220
XIV-A. System Functions in the G.C.D. - Newton's Method to be Typed When the WATFOR Compiler is Used	221
XV. System Functions for Converting Single Precision G.C.D. - Newton's Method to Double Precision	222
XVI. Control Data for G.C.D. - Newton's Method	223
XVII. Correspondence of Newton's and G.C.D. - Newton's Input Data Variables	224
XVIII. Variables Used in the G.C.D. - Newton's Method	228
XVIII-A. Single Precision Program for G.C.D. - Newton's Method	242
XVIII-B. Double Precision Program for G.C.D. - Newton's Method	254
XIX. Program Changes for Solving Polynomials of Degree Greater Than 25 by G.C.D. - Muller's Method	269
XX. System Functions for Converting Single Precision G.C.D. - Muller's Method to Double Precision	270
XXI. Variables Used in G.C.D. - Muller's Method	271
XXI-A. Single Precision Program for G.C.D. - Muller's Method	282
XXI-B. Double Precision Program for G.C.D. - Muller's Method	296
XXII. Arrays Used in the Polynomial Generator	313
XXIII. Control Data for the Polynomial Generator	314
XXIV. Factor Data for the Polynomial Generator	315
XXV. Data to End Execution of the Polynomial Generator	316
XXVI. Variables Used in the Polynomial Generator	321
XXVII. Single Precision Program for the Polynomial Generator	324
XXVIII. Double Precision Program for the Polynomial Generator	325

LIST OF FIGURES

Figure	Page
1. Geometrical Interpretation of Newton's Method	12
2. Geometrical Interpretation of Muller's Method	22
2.1. Covering on Annulus With Eight Overlapping Circles	37
3. Geometric Interpretation of Lehmer's Method	40
3.1. The Q-D Scheme	52
3.2. Starting the Q-D Scheme	54
3.3. Scheme for the Polynomials of Theorem 6.3	68
3.4. Scheme for Removing Zero Coefficients	70
4. Generating Initial Approximations	128
5. Altering Approximations	129
6. Distribution of Approximations	130
7. Sequence of Input Data for Newton's Method	144
8. Control Card for Newton's Method	145
9. Coefficient Card for Newton's Method	146
10. Initial Approximation Card for Newton's Method	147
11. End Card for Newton's Method	147
12. Flow Charts for Newton's Method	157
12.1. Flow Charts for Muller's Method	191
13. Control Card for G.C.D. - Newton's Method	225
13.1. Flow Charts for G.C.D. - Newton's Method	232
13.2. Flow Charts for G.C.D. - Muller's Method	272
14. Control Card for the Polynomial Generator	318

Figure	Page
15. Factor Card for the Polynomial Generator	319
16. End Card for the Polynomial Generator	320
17. Sequence of Input Data for the Polynomial Generator	320
18. Flow Charts for the Polynomial Generator	323

LIST OF EXHIBITS

Exhibit	Page
A.	88
B.	90
C.	92
D.	94
E.	96
F.	97
G.	98
H.	99
I.	100
J.	101
K.	102
L.	103
M.	104
N.	105
O.	107
P.	108
Q.	109
R.	110
S.	111
T.	112
U.	113
V.	114

Exhibit	Page
W.	115
X.	116
Y.	117
Z.	118
AA.	131
BB.	132
CC.	133
DD.	322

CHAPTER I

INTRODUCTION

Frequently in scientific work it becomes necessary to find the zeros, real or complex, of the polynomial of degree N

$$P(X) = a_1 X^N + a_2 X^{N-1} + \dots + a_N X + a_{N+1}$$

where $a_1 \neq 0$, and the coefficients a_1, a_2, \dots, a_{N+1} are complex numbers. Various classical methods calculate the exact roots of polynomials of degree 1, 2, 3, or 4. For polynomials of higher degree, no such methods exist. Thus, to solve for the zeros of such polynomials, numerical methods of iteration based on successive approximations must be employed. In the following material five such methods are given which are particularly suited for modern high speed computers.

Newton's method is an iterative procedure which generates a sequence of successive approximations of a zero of $P(X)$ by using the iteration formula

$$X_{n+1} = X_n - P(X_n)/P'(X_n).$$

An initial approximation to the zero is required to start the iterative process. Under certain conditions this sequence will converge quadratically to the desired root. It is, however, necessary to compute the value of the polynomial and its derivative for each step in the

iterative procedure. Once a zero of $P(X)$ has been found, it is divided out of $P(X)$, giving a deflated polynomial of lower degree. $P(X)$ is replaced by the deflated polynomial and the iterative process is applied to extract another zero of $P(X)$. This procedure is repeated until all zeros of $P(X)$ have been found. The zeros may then be rechecked and their accuracy possibly improved by using them as initial approximations with Newton's process applied to the full (undeflated) polynomial.

Muller's method is also an iterative procedure generating a sequence $X_1, X_2, \dots, X_n, \dots$ of successive approximations of a root of $P(X)$. This method converges almost quadratically near a zero and does not require the evaluation of the derivative of the polynomial. Muller's method requires three distinct approximations of a root to start the process of iteration. A quadratic equation is constructed through the three given points as an approximation of $P(X)$. The root of the quadratic closest to X_n is taken as X_{n+1} , the next approximation to the zero. This process is then repeated on the last three points of the sequence. After a root of $P(X)$ has been found, $P(X)$ is deflated, and replaced in the above procedure by the deflated polynomial. After all zeros of $P(X)$ are found from successive deflations, they are improved as in Newton's method.

The greatest common divisor method reduces the problem of finding all zeros (possibly multiple zeros) of $P(X)$ to one of extracting the zeros of a polynomial $P_1(X) = P(X)/D(X)$, all of whose zeros are simple. $D(X)$, the greatest common divisor of $P(X)$ and its derivative, $P'(X)$, is obtained by repeated application of the division algorithm. Once $P_1(X)$ is obtained, some suitable method such as Newton's or Muller's method

is used to find the zeros of $P_1(X)$. By finding all the zeros of $P_1(X)$, all the zeros of $P(X)$ are obtained. The multiplicity of each zero may then be determined.

Lehmer's method locates zeros of the polynomial $P(X)$ by searching the complex plane. This method repeatedly seeks to answer the question "Does $P(X)$ have a zero inside a given circle?" The procedure is started with the unit circle. The radius is repeatedly halved (or doubled) until an annulus is found which contains a root while the inner circle is free of roots. The annulus can be covered by eight overlapping circles. The above question is asked of each circle until one is found containing a root. From the center of this circle the above process is repeated. Continuing, we obtain a circle of sufficiently small radius about the root. When a root has been found, it is divided out of $P(X)$ by deflation and the method repeated on the deflated polynomial to extract another root. Lehmer's method applied to the successively deflated polynomials gives all the roots of $P(X)$.

The quotient-difference method is an iterative procedure producing, simultaneously, approximations to all roots of $P(X)$. The zeros are extracted roughly in order of decreasing magnitude. No initial approximations are necessary for use of this method. Convergence of this method is somewhat slower than other methods and substantial round-off error can occur.

In the material to follow, an individual examination of each method will be presented followed by a conclusion containing a comparison of the methods. These examinations will include the algorithms necessary to employ the method together with the conditions necessary for convergence. The conclusion will include advantages and

disadvantages of each method and a discussion concerning the extraction of zeros of ill-conditioned polynomials. An appendix for each method will present the flow diagrams, listing of the program, definition of variables used in the program, and instructions for use of the program.

CHAPTER II

NEWTON'S METHOD

1. Derivation of the Algorithm

Newton's method is probably the most popular iterative procedure for finding the zeros of a polynomial. This fact is due to the excellent results obtained, the simplicity of the computational routine, and the fast rate of convergence obtained provided the initial approximation of a zero is close enough. Also, the method can be applied to the extraction of complex as well as real zeros.

Consider the polynomial

$$P(X) = a_1 X^N + a_2 X^{N-1} + \dots + a_N X + a_{N+1} \quad (2-1)$$

where $a_1 \neq 0$ and the coefficients a_1, a_2, \dots, a_{N+1} are complex. The algorithm for Newton's method can be derived by approximating $P(X)$ by a Taylor series expansion about an approximation, X_0 , of a zero, α , of $P(X)$. Using only the first two terms of the expansion, the expression

$$P(X) \doteq P(X_0) + P'(X_0)(X - X_0)$$

is obtained. If this equation is solved for $P(X) = 0$, then

$$0 \doteq P(X_0) + P'(X_0)(X - X_0)$$

results. Rearranging terms produces

$$0 \doteq P(X_0) + P'(X_0) X - P'(X_0) X_0$$

followed by

$$P'(X_0) X_0 - P(X_0) \doteq P'(X_0) X$$

from which division by $P'(X_0)$ produces

$$X_0 - P(X_0)/P'(X_0) \doteq X$$

which is the basic formula for Newton's method. Thus, in general, we obtain the $(n+1)^{\text{th}}$ approximation, X_{n+1} , of α from the n^{th} approximation, X_n , by

$$X_{n+1} = X_n - P(X_n)/P'(X_n). \quad (2-2)$$

As a result of repeated use of this algorithm, we obtain the sequence

$$X_0, X_1, X_2, \dots, X_n, \dots \quad (2-3)$$

of successive approximations of the root, α . It should be noted that an initial approximation is necessary to start the iterative process for each new zero; that is, a polynomial of degree N may require N initial approximations.

In order to use equation (2-2), it is necessary to compute, for each X_n , the value of the polynomial, $P(X_n)$, and its derivative, $P'(X_n)$. The division algorithm states that if $P(X)$ and $G(X)$ are polynomials, then there exists polynomials $H(X)$ and $K(X)$ such that $P(X) = H(X)G(X) + K(X)$ where $K(X) = 0$ or $\text{deg. } K(X) < \text{deg. } G(X)$. From this expression of $P(X)$, the following remainder theorem is obtained:

Theorem 2.1. If $P(X)$ is a polynomial and c is a complex number, then the remainder obtained from dividing $P(X)$ by $(X - c)$ is $P(c)$.

The proof of Theorem 2.1 is given in [6, P. 102]. Thus, $P(X)$ can be written as $P(X) = (X - c) H(X) + R$ where $P(c) = R$. $P'(X)$ is then obtained by the following theorem, the proof of which can be found in [6, PP. 105-106].

Theorem 2.2. If $P(X)$ and $H(X)$ are polynomials and c is a complex number such that $P(X) = (X - c) H(X) + R$ where $P(c) = R$, then the remainder obtained from dividing $H(X)$ by $(X - c)$ is $P'(c)$.

From synthetic division, an algorithm known as Horner's Method is acquired for computing $P(X_n)$ and $P'(X_n)$.

Theorem 2.3. Let $P(X)$ be defined as in equation (2-1) and let d be a complex number. Define a sequence b_1, b_2, \dots, b_{N+1} by

$$b_1 = a_1$$

$$b_i = a_i + db_{i-1} \quad (i = 2, 3, \dots, N+1).$$

Define another sequence c_1, c_2, \dots, c_N by

$$c_1 = b_1$$

$$c_j = b_j + dc_{j-1} \quad (j = 2, 3, \dots, N).$$

Then $P(d) = b_{N+1}$ and $P'(d) = c_N$. The elements b_1, b_2, \dots, b_N are the coefficients of the polynomial $H(X)$ in Theorem 2.2 when $P(X)$ is divided by $(X - d)$.

These formulas are derived in [6, PP. 106-107]. Thus with equation (2-2) and the iteration formulas of the previous theorem, Newton's method can now be applied to generate the sequence (2-3) which will converge to the root, α , if the convergence conditions given in Theorem 2.4 are satisfied.

A criterion is needed to determine when to terminate the sequence (2-3); that is, when has a zero been found? For convergence of the sequence, there must exist a term in the sequence beyond which the difference between any two successive terms is arbitrarily small. Therefore, it is desirable to make the quotient $|X_n/X_{n+1}|$ sufficiently near 1. From equation (2-2)

$$1 = \left| \frac{X_n}{X_{n+1}} - \frac{\frac{P(X_n)}{P'(X_n)}}{X_{n+1}} \right|$$

$$\leq \left| \frac{X_n}{X_{n+1}} \right| + \left| \frac{\frac{P(X_n)}{P'(X_n)}}{X_{n+1}} \right|$$

Thus

$$1 + \left| \frac{\frac{P(X_n)}{P'(X_n)}}{X_{n+1}} \right| \leq \left| \frac{X_n}{X_{n+1}} \right|$$

where $P'(X_n)$ and $X_{n+1} \neq 0$. Thus, iterations are continued until an X_n is obtained such that $|P(X_n)/P'(X_n)|/|X_{n+1}|$ is as small as desired.

After a zero, α , of $P(X)$ has been found, the term $(X - \alpha)$ is synthetically divided out of $P(X)$ by deflation using Theorem 2.3 obtaining

a polynomial, $P_1(X)$, of degree $N-1$. The root finding process is then repeated to extract a zero, α_1 , of $P_1(X)$. $P(X)$ can be written as

$$P(X) = (X - \alpha) P_1(X) + R$$

where $R = P(\alpha)$. But $P(\alpha) = 0$. Therefore, substitution produces

$$P(X) = (X - \alpha) P_1(X).$$

Now $P_1(\alpha_1) = 0$ implies that $P(\alpha_1) = 0$. Hence, α_1 is a zero of $P(X)$.

By the process of root finding and successive deflations, zeros $\alpha_0, \alpha_1, \dots, \alpha_{N-1}$ of the deflated polynomials

$$P(X) = P_0(X), P_1(X), \dots, P_{N-1}(X),$$

respectively, are extracted. Each α_i ($i = 0, 1, 2, \dots, N-1$) is a zero of $P(X)$ since each α_i is a zero of $P_{i-1}(X), P_{i-2}(X), \dots, P_1(X), P(X)$.

After all zeros of $P(X)$ have been found, it may be possible to improve their accuracy by using them as initial approximations with Newton's method applied to the full (undeflated) polynomial, $P(X)$. This should correct any loss of accuracy which may have resulted from the successive deflations.

2. Convergence of Newton's Method

The following theorem from [3, PP. 79-81] gives sufficient conditions for the convergence of sequence (2-3).

Theorem 2.4. Let $P(X)$ be a polynomial and let the following conditions be satisfied on the closed interval $[a, b]$:

1. $P(a)P(b) < 0$
2. $P'(X) \neq 0, X \in [a,b]$
3. $P''(X)$ is either ≥ 0 or ≤ 0 for all $X \in [a,b]$
4. If c denotes the endpoint of $[a,b]$ at which $|P'(X)|$ is smaller, then $|P(c)/P'(c)| \leq b - a$.

Then Newton's method converges to the (only) solution, s , of $P(X) = 0$ for any choice of X_0 in $[a,b]$.

When convergence is obtained, it is quadratic; that is,

$$e_{i+1} = \frac{1}{2} F''(\eta_i) e_i^2$$

where $F(X_i) = X_i - P(X_i)/P'(X_i)$, η_i is between X_i and the zero, α , and e_i is the error in X_i . This means that the error obtained in the $(i+1)^{\text{th}}$ iteration of Newton's algorithm is proportional to the square of the error obtained in the i^{th} iteration. A proof of quadratic convergence can be found in [2, PP. 31-33].

3. Procedure for Newton's Method

The general procedure for applying Newton's method is enumerated sequentially as follows, starting with initial approximation X_0 :

1. Calculate a new approximation X_{n+1} by

$$X_{n+1} = X_n - P(X_n)/P'(X_n).$$

2. Test for convergence; that is, test

$$\left| \frac{P(X_n)/P'(X_n)}{X_{n+1}} \right| < \epsilon$$

for some ϵ chosen as small as desired.

3. If convergence is obtained, perform the following:

- a. Save X_{n+1} as the desired approximation to a zero of $P(X)$.
 - b. Deflate $P(X)$ using X_{n+1} .
 - c. Replace $P(X)$ by the deflated polynomial.
 - d. Return to step 1 with a new initial approximation.
4. If no convergence is obtained, increase n by 1 and return to step 1.

In order to prevent an unending iteration process in case the method does not produce convergence, a maximum number of iterations should be specified. If convergence is not obtained within this number of iterations, change the initial approximation and return to step 1 above.

4. Geometrical Interpretation of Newton's Method

A geometrical interpretation of Newton's method is given in Figure 1. X_i is an approximation to the zero, α . $P'(X_i)$ is the slope of the line tangent to $P(X)$ at X_i . X_{i+1} is the intersection of the tangent line with the x axis.

5. Determining Multiple Roots

If $P(X)$ has m distinct zeros, then $P(X)$ can be written as

$$P(X) = a_1 (X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \dots (X - \alpha_m)^{e_m}, \quad (m \leq N)$$

where α_i is a zero of $P(X)$ and e_i is the multiplicity of α_i ($i = 1, 2, \dots, m$). Consider the root α_j . Dividing out the term

$(X - \alpha_j)$ by deflating $P(X)$ gives $P_1(X)$ of degree $N-1$ which can be written as

$$P_1(X) = (X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \dots (X - \alpha_j)^{e_j-1} \dots (X - \alpha_m)^{e_m}.$$

Evaluating $P_1(X)$ at the zero, α_j , gives $P_1(\alpha_j) = 0$ if $e_j > 1$. Thus, after a zero, α , of $P(X)$ is determined by Newton's iterative process and the current polynomial is deflated giving $P_1(X)$, then $P_1(\alpha)$ is evaluated. If $P_1(\alpha) \leq \epsilon$ for some small number ϵ , α is a root of $P_1(X)$ and thus has multiplicity at least equal to two. $P_1(X)$ is then deflated giving $P_2(X)$. If $P_2(\alpha) \leq \epsilon$, α is of multiplicity at least three. This process is continued until a deflated polynomial $P_k(X)$ is encountered such that either $\deg. P_k(X) = 0$ or $P_k(\alpha) > \epsilon$. α is then a zero of multiplicity $k+1$.

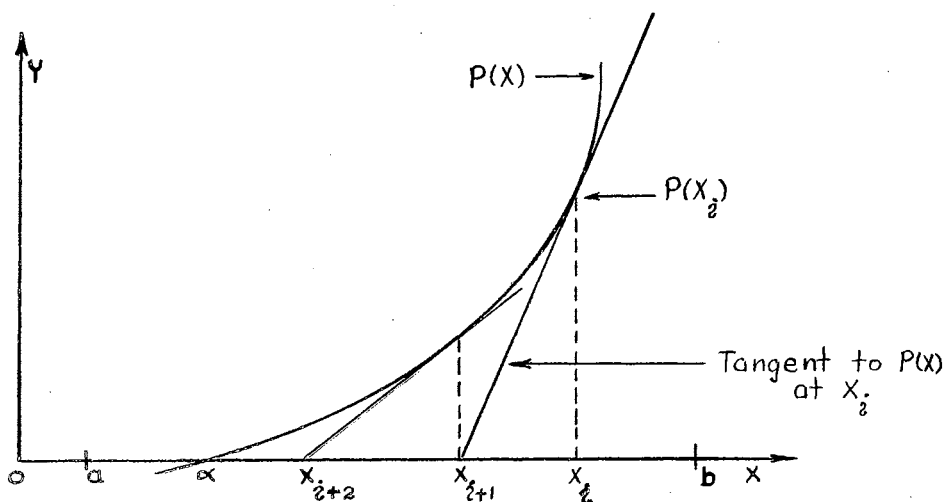


Figure 1. Geometrical Interpretation of Newton's Method

CHAPTER III

MULLER'S METHOD

1. Derivation of the Algorithm

Muller's method in [5] is an iterative procedure designed to find any prescribed number of zeros, real or complex, of a polynomial. The method does not require the evaluation of the derivative and near a zero the convergence is almost quadratic.

Consider the polynomial

$$P(X) = a_1 X^N + a_2 X^{N-1} + \dots + a_N X + a_{N+1} \quad (3-1)$$

with complex coefficients such that $a_1 \neq 0$. Given three distinct approximations, X_{n-2}, X_{n-1}, X_n , to a root, α , of $P(X)$, the problem is to determine X_{n+1} in such a way as to generate a sequence

$$X_1, X_2, X_3, \dots, X_n, X_{n+1}, \dots \quad (3-2)$$

of approximations converging to α . The points $(X_{n-2}, P(X_{n-2}))$, $(X_{n-1}, P(X_{n-1}))$, and $(X_n, P(X_n))$ determine a unique quadratic polynomial, $Q(X)$, approximating $P(X)$ in the vicinity of X_{n-2}, X_{n-1}, X_n . A general proof of this can be found in [3, PP. 133-134]. Thus, the zeros of $Q(X)$ will be approximations of the zeros of $P(X)$ in this region of approximation. From the general representation in [3, P. 184] of the Lagrangian interpolating polynomial, the representation of $Q(X)$ is given by

$$\begin{aligned}
Q(X) &= \frac{(X - X_{n-1})(X - X_{n-2})}{(X_n - X_{n-1})(X_n - X_{n-2})} P(X_n) \\
&+ \frac{(X - X_n)(X - X_{n-2})}{(X_{n-1} - X_n)(X_{n-1} - X_{n-2})} P(X_{n-1}) \\
&+ \frac{(X - X_n)(X - X_{n-1})}{(X_{n-2} - X_n)(X_{n-2} - X_{n-1})} P(X_{n-2})
\end{aligned}$$

which can be rewritten as

$$\begin{aligned}
Q(X) &= Q(X - X_n + X_n) \\
&= \frac{(X - X_n + X_n - X_{n-1})(X - X_n + X_n - X_{n-1} + X_{n-1} - X_{n-2})}{(X_n - X_{n-1})(X_n - X_{n-1} + X_{n-1} - X_{n-2})} P(X_n) \\
&- \frac{(X - X_n)(X - X_n + X_n - X_{n-1} + X_{n-1} - X_{n-2})}{(X_n - X_{n-1})(X_{n-1} - X_{n-2})} P(X_{n-1}) \\
&+ \frac{(X - X_n)(X - X_n + X_n - X_{n-1})}{(X_n - X_{n-1} + X_{n-1} - X_{n-2})(X_{n-1} - X_{n-2})} P(X_{n-2}).
\end{aligned}$$

In order to simplify this expression, introduce the quantities

$$h_n = X_n - X_{n-1}, \quad h = X - X_n.$$

Then

$$\begin{aligned}
Q(X) &= Q(X_n + h) \\
&= \frac{(h + h_n)(h + h_n + h_{n-1})}{h_n(h_n + h_{n-1})} P(X_n) \\
&- \frac{h(h + h_n + h_{n-1})}{h_n h_{n-1}} P(X_{n-1})
\end{aligned}$$

$$\begin{aligned}
& + \frac{h(h+h_n)}{(h_n+h_{n-1})h_{n-1}} P(X_{n-2}) \\
& = \frac{h^2 + 2hh_n + hh_{n-1} + h_n^2 + h_n h_{n-1}}{h_n^2 + h_n h_{n-1}} P(X_n) \\
& \quad - \frac{h^2 + hh_n + hh_{n-1}}{h_n h_{n-1}} P(X_{n-1}) \\
& \quad + \frac{h^2 + hh_n}{h_n h_{n-1} + h_{n-1}^2} P(X_{n-2}).
\end{aligned}$$

Collecting terms containing like powers of h produces

$$Q(X) = Q(X_n + h)$$

$$\begin{aligned}
& = \left(\frac{P(X_n)}{h_n^2 + h_n h_{n-1}} - \frac{P(X_{n-1})}{h_n h_{n-1}} + \frac{P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h^2 \\
& + \left(\frac{(2h_n + h_{n-1}) P(X_n)}{h_n^2 + h_n h_{n-1}} - \frac{(h_n + h_{n-1}) P(X_{n-1})}{h_n h_{n-1}} + \frac{h_n P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h \\
& + \frac{h_n (h_n + h_{n-1}) P(X_n)}{h_n^2 + h_n h_{n-1}} \\
& = \left(\frac{P(X_n) h_{n-1}}{h_n^2 h_{n-1} + h_n h_{n-1}^2} - \frac{P(X_{n-1})}{h_n h_{n-1}} + \frac{P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h^2 \\
& + \left(\frac{(2h_n h_{n-1} + h_{n-1}^2) P(X_n)}{h_n^2 h_{n-1} + h_n h_{n-1}^2} - \frac{(h_n + h_{n-1}) P(X_{n-1})}{h_n h_{n-1}} + \frac{h_n P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h
\end{aligned}$$

$$+ \frac{(h_n^2 h_{n-1} + h_{n-1}^2 h_n) P(X_n)}{h_n^2 h_{n-1} + h_n h_{n-1}^2}$$

Using the common denominator, $h_n^2 h_{n-1} + h_n h_{n-1}^2$, and combining terms yields

$$Q(X_n + h) = \left(\frac{P(X_n) h_{n-1} - P(X_{n-1})(h_n + h_{n-1}) + P(X_{n-2}) h_n}{h_n^2 h_{n-1} + h_n h_{n-1}^2} \right) h^2$$

$$+ \left(\frac{(2h_n h_{n-1} + h_{n-1}^2) P(X_n) - (h_n + h_{n-1})^2 P(X_{n-1}) + h_n^2 P(X_{n-2})}{h_n^2 h_{n-1} + h_n h_{n-1}^2} \right) h$$

$$+ \frac{(h_n^2 h_{n-1} + h_{n-1}^2 h_n) P(X_n)}{h_n^2 h_{n-1} + h_n h_{n-1}^2}$$

Multiplying by h_n/h_{n-1}^2 results in

$$Q(X_n + h) = \left[\frac{P(X_n) \frac{h_n}{h_{n-1}} - P(X_{n-1}) \left(\left(\frac{h_n}{h_{n-1}} \right)^2 + \frac{h_n}{h_{n-1}} \right) + P(X_{n-2}) \left(\frac{h_n}{h_{n-1}} \right)^2}{\frac{h_n^3}{h_{n-1}} + h_n^2} \right] h^2$$

$$+ \left[\frac{\left(\frac{h_n^2}{h_{n-1}} + h_n \right) P(X_n) - h_n \left[\left(\frac{h_n}{h_{n-1}} \right) + \left(\frac{h_{n-1}}{h_{n-1}} \right) \right]^2 P(X_{n-1}) + \frac{h_n^3}{h_{n-1}^2} P(X_{n-2})}{\frac{h_n^3}{h_{n-1}} + h_n^2} \right] h$$

$$+ \left[\frac{\left(\frac{h_n^3}{h_{n-1}} + h_n^2 \right) P(X_n)}{\frac{h_n^3}{h_{n-1}} + h_n^2} \right]$$

Let $q_n = \frac{h_n}{h_{n-1}}$ and $q = \frac{h}{h_n}$. Then

$$Q(X_n + h) = \left(\frac{P(X_n) q_n - P(X_{n-1})(q_n^2 + q_n) + P(X_{n-2}) q_n^2}{q_n + 1} \right) q^2$$

$$+ \left(\frac{(2q_n + 1) P(X_n) - (q_n + 1)^2 P(X_{n-1}) + q_n^2 P(X_{n-2})}{q_n + 1} \right) q$$

$$+ \frac{(q_n + 1) P(X_n)}{q_n + 1}$$

Now let

$$A_n = q_n P(X_n) - q_n (q_n + 1) P(X_{n-1}) + q_n^2 P(X_{n-2})$$

$$B_n = (2q_n + 1) P(X_n) - (q_n + 1)^2 P(X_{n-1}) + q_n^2 P(X_{n-2})$$

$$C_n = (q_n + 1) P(X_n)$$

Then

$$Q(X_n + h) = Q(X_n + qh_n)$$

and

$$Q(X_n + qh_n) = \frac{A_n q^2 + B_n q + C_n}{q_n + 1}$$

Solving the quadratic equation $Q(X_n + qh_n) = 0$ and denoting the result by q_{n+1} gives:

$$q_{n+1} = \frac{-B_n \pm \sqrt{B_n^2 - 4A_n C_n}}{2A_n}$$

and the new approximation is found as follows:

$$q_{n+1} = \frac{h_{n+1}}{h_n} = \frac{X_{n+1} - X_n}{h_n}$$

Thus

$$X_{n+1} = X_n + h_n q_{n+1}$$

In order to avoid loss of accuracy, q_{n+1} can be written in a better form as follows:

$$\begin{aligned} q_{n+1} &= \frac{-B_n \pm \sqrt{B_n^2 - 4A_n C_n}}{2A_n} \cdot \frac{B_n \pm \sqrt{B_n^2 - 4A_n C_n}}{B_n \pm \sqrt{B_n^2 - 4A_n C_n}} \\ &= \frac{-B_n^2 \pm B_n^2 - 4A_n C_n}{2A_n (B_n \pm \sqrt{B_n^2 - 4A_n C_n})} \\ q_{n+1} &= \frac{-2C_n}{B_n \pm \sqrt{B_n^2 - 4A_n C_n}} \end{aligned} \quad (3-3)$$

The sign in the denominator should be chosen such that the magnitude of the denominator is largest, thus causing $|q_{n+1}|$ to be smallest. This, in turn, will make X_{n+1} closest to X_n .

Note that each iteration of this process requires three approximations, X_{n-2}, X_{n-1}, X_n , in order to compute X_{n+1} . Thus, when X_{n+1} is found, X_{n-1}, X_n, X_{n+1} are used to compute X_{n+2} ; that is, the last three terms of the generated sequence are used to compute the next term.

Convergence of the sequence (3-2) to a zero is obtained when the elements X_k and X_{k+1} of the sequence are found such that

$$\frac{|X_{k+1} - X_k|}{|X_{k+1}|} < \epsilon, \quad X_{k+1} \neq 0;$$

that is, the ratio of the change in the approximation to the approximation itself is as small as desired.

In order to use the iterative formulas, it is necessary to compute the value, $P(X_j)$, of the polynomial $P(X)$ at the approximation X_j . The procedure for doing this is discussed in Chapter II, § 1. The iteration formulas are given in Theorem 2.3 of Chapter II.

After a zero, α , of $P(X)$ has been found, $P(X)$ is deflated as described in Chapter II, § 1, and the process repeated to extract a zero, α_1 , of $P_1(X)$. By applying Muller's method to successively deflated polynomials, all the zeros of $P(X)$ are obtained. For more detailed discussion of this procedure see Chapter II, § 1, keeping in mind that Muller's instead of Newton's method is used.

Muller's method requires three initial approximations to a zero in order to start the iteration process. If three are not known, the values $X_1 = -1, X_2 = 1, X_3 = 0$ can be used.

Convergence of Muller's method is almost quadratic provided the three initial approximations are sufficiently close to a zero of $P(X)$. This is natural to expect since $P(X)$ is being approximated by a

quadratic polynomial. Quadratic convergence means that the error obtained in the $(n+1)^{\text{th}}$ step of the iterative process is proportional to the square of the error obtained in the n^{th} iteration. However, no general proof of convergence has been obtained for Muller's method. It has produced convergence in the majority of the cases tested.

In application of Muller's method, an alteration should be made to handle the case in which the denominator of equation (3-3) is zero (0). This occurs whenever $P(X_n) = P(X_{n-1}) = P(X_{n-2})$. If this happens, set $q_{n+1} = 1$.

Another alteration which should be made in actual practice is to compute the quantity $|P(X_{n+1})| / |P(X_n)|$ whenever the value $P(X_{n+1})$ is calculated. If the former quantity exceeds ten (10), q_{n+1} is halved and h_n , X_{n+1} , and $P(X_{n+1})$ are recomputed accordingly.

2. Procedure for Muller's Method

The basic steps performed by Muller's method are listed sequentially as follows, starting with initial approximations X_1 , X_2 , and X_3 .

1. Compute h_n , q_n , D_n , B_n , C_n , q_{n+1} as defined previously.
2. Compute the next approximation X_{n+1} by

$$X_{n+1} = X_n + h_n q_{n+1}.$$

3. Test for convergence; that is, test

$$\left| \frac{X_{n+1} - X_n}{X_{n+1}} \right| < \epsilon$$

for some suitably small number ϵ .

4. If the test fails, return to step 1 with the last three approximations X_{n+1} , X_n , X_{n-1} .

5. If the test passes, do the following:
 - a. Save X_{n+1} as the desired approximation to a zero.
 - b. Deflate the current polynomial using X_{n+1} .
 - c. Replace the current polynomial by the deflated polynomial.
 - d. Return to step 1 with a new set of initial approximations.

In order to avoid an unending iteration process in case the method does not produce convergence, a maximum number of iterations should be specified. If convergence is not obtained within this number of iterations, the initial approximations should be altered.

3. Geometrical Interpretation of Muller's Method

Figure 2 shows the geometrical interpretation of Muller's method for real roots of $P(X)$ and the quadratic $Q(X)$. The root of $Q(X)$ closest to X_i is chosen as the next approximation X_{i+1} .

4. Determining Multiple Roots

For a discussion concerning multiple roots see Chapter II, § 5.

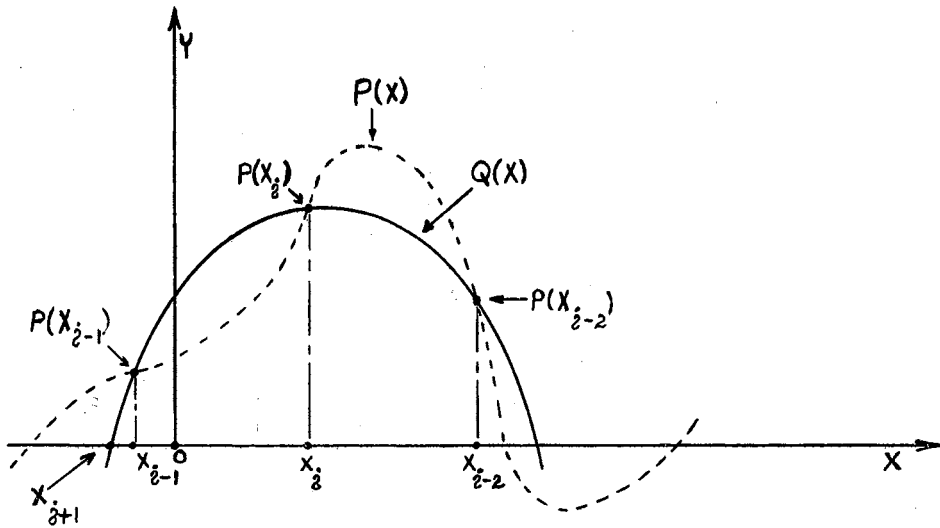


Figure 2. Geometrical Interpretation of Muller's Method

CHAPTER IV

GREATEST COMMON DIVISOR METHOD

1. Derivation of the Algorithm

The greatest common divisor (g.c.d.) method reduces the problem of finding all the zeros of a polynomial, possibly having multiple zeros, to one of solving for zeros of a polynomial all of whose zeros are simple.

Consider the N^{th} degree polynomial

$$P(X) = a_1 X^N + a_2 X^{N-1} + \dots + a_N X + a_{N+1} \quad (4-1)$$

where $a_1 \neq 0$ and a_1, a_2, \dots, a_{N+1} are complex numbers. If $P(X)$ has m distinct zeros, $\alpha_1, \alpha_2, \dots, \alpha_m$, then $P(X)$ can be expressed in the form

$$P(X) = a_1 (X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \dots (X - \alpha_m)^{e_m}$$

where e_i is the multiplicity of α_i , $i = 1, 2, \dots, m$. The derivative of $P(X)$ is

$$P'(X) = N a_1 X^{N-1} + (N-1) a_2 X^{N-2} + \dots + a_N$$

which can also be expressed as

$$P'(X) = a_1 (X - \alpha_1)^{e_1-1} (X - \alpha_2)^{e_2-1} \dots (X - \alpha_m)^{e_m-1} \sum_{i=1}^m e_i \prod_{\substack{j=1 \\ j \neq i}}^m (X - \alpha_j).$$

(4-2)

The greatest common divisor of $P(X)$ and $P'(X)$ is obtained from the following theorem.

Theorem 4.1. Let $P(X)$ be an N^{th} degree polynomial having m distinct zeros $\alpha_1, \alpha_2, \dots, \alpha_m$ of multiplicity e_1, e_2, \dots, e_m respectively. Then the polynomial

$$D(X) = (X - \alpha_1)^{e_1-1} (X - \alpha_2)^{e_2-1} \dots (X - \alpha_m)^{e_m-1}$$

is the unique monic greatest common divisor of $P(X)$ and its derivative $P'(X)$.

Proof. It suffices to show the following:

- a. $D(X)$ divides $P(X)$
- b. $D(X)$ divides $P'(X)$
- c. If $K(X)$ is a polynomial such that $K(X)$ divides both $P(X)$ and $P'(X)$, then $K(X)$ divides $D(X)$.

$P'(X)$ can be written as

$$P'(X) = D(X) S(X) \text{ where } S(X) = \sum_{i=1}^m e_i \prod_{\substack{j=1 \\ j \neq i}}^m (X - \alpha_j).$$

$D(X)$ divides $P(X)$ since

$$P(X) = D(X) \prod_{i=1}^m (X - \alpha_i).$$

$D(X)$ divides $P'(X)$ because

$$P'(X) = D(X) \sum_{i=1}^m e_i \prod_{\substack{j=1 \\ j \neq i}}^m (X - \alpha_j).$$

Both the polynomials $P(X)$ and $P'(X)$ can be expressed uniquely as a constant times a product of monic irreducible polynomials. A general proof of this can be found in [7, P. 121]. Note that even though there are e_i of the factors $(X - \alpha_i)$ in $P(X)$ and $e_i - 1$ of them in $P'(X)$, we will consider them distinct in the sense that they are not identical. Let $K(X)$ be a non-constant polynomial such that $K(X)$ divides $P(X)$ and $P'(X)$. Without loss of generality we may assume that $K(X)$ is monic. Since $K(X)$ divides $P(X)$, then there exists a polynomial $T(X)$ such that

$$P(X) = K(X) T(X). \quad (4-3)$$

From Theorem 3.35 of [7, P. 121], $K(X)$ and $T(X)$ can be expressed in the forms:

$$K(X) = (X - \beta_1)^{g_1} (X - \beta_2)^{g_2} \dots (X - \beta_v)^{g_v}$$

$$T(X) = b(X - \eta_1)^{f_1} (X - \eta_2)^{f_2} \dots (X - \eta_u)^{f_u}$$

where the factors are unique apart from order. Substituting for $P(X)$, $T(X)$, and $K(X)$ in equation (4-3) gives

$$\begin{aligned}
 & a_1 (X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \dots (X - \alpha_m)^{e_m} \\
 & = [(X - \beta_1)^{g_1} \dots (X - \beta_v)^{g_v}] [b(X - \eta_1)^{f_1} \dots (X - \eta_u)^{f_u}]; \quad (4-4)
 \end{aligned}$$

that is, both sides of equation (4-4) are unique representations of $P(X)$. After proper ordering of the terms on the right hand side of equation (4-4), we conclude from the uniqueness of the two expressions that each factor of $K(X)$ must be a factor of $P(X)$. This implies that $X - \beta_i = X - \alpha_i$ for $i = 1, \dots, v$. Also, $g_i \leq e_i$ for $i = 1, 2, \dots, v$ since $K(X)$ cannot contain a greater number of any one factor than $P(X)$. Finally, $v \leq m$ since $K(X)$ cannot have more distinct factors than $P(X)$.

Thus

$$K(X) = (X - \alpha_1)^{g_1} (X - \alpha_2)^{g_2} \dots (X - \alpha_v)^{g_v}. \quad (4-5)$$

Since $K(X)$ divides $P'(X)$, there exists a polynomial $R(X)$ such that

$$P'(X) = K(X) R(X). \quad (4-6)$$

Again, $R(X)$ can be expressed uniquely as

$$R(X) = d(X - \lambda_1)^{h_1} (X - \lambda_2)^{h_2} \dots (X - \lambda_t)^{h_t}.$$

Also, $P'(X)$, given in equation (4-2), can be expressed as

$$\begin{aligned}
 P'(X) & = [a_1 (X - \alpha_1)^{e_1-1} (X - \alpha_2)^{e_2-1} \dots (X - \alpha_m)^{e_m-1}] \\
 & \quad [c(X - \delta_1)^{e_{m+1}} \dots (X - \delta_w)^{e_{m+w}}]
 \end{aligned}$$

since

$$\sum_{i=1}^m e_i \prod_{\substack{j=1 \\ j \neq i}}^m (X - \alpha_j)$$

is a polynomial and can be expressed as a constant times a product of unique linear factors. Thus, substituting for $P'(X)$, $K(X)$, and $R(X)$ of equation (4-6) gives

$$\begin{aligned} & [a_1(X - \alpha_1)^{e_1-1} \dots (X - \alpha_m)^{e_m-1}] [c(X - \delta_1)^{e_{m+1}} \dots (X - \delta_w)^{e_{m+w}}] \\ &= [(X - \alpha_1)^{g_1} \dots (X - \alpha_v)^{g_v}] [d(X - \lambda_1)^{h_1} \dots (X - \lambda_t)^{h_t}]. \end{aligned}$$

Again, since both sides are unique representations of $P'(X)$, then each factor $X - \alpha_i$ must be contained in $P'(X)$ for $i = 1, \dots, v$. But $X - \alpha_i$ is not in the product

$$\prod_{\substack{j=1 \\ j \neq i}}^m (X - \alpha_j)$$

and is, therefore, not a factor of $c(X - \delta_1)^{e_{m+1}} \dots (X - \delta_w)^{e_{m+w}}$. Hence, $X - \alpha_i$ is contained in $a_1(X - \alpha_1)^{e_1-1} \dots (X - \alpha_m)^{e_m-1}$ for $i = 1, \dots, v$.

Thus, $g_i \leq e_i - 1$, $i = 1, 2, \dots, v$ since $K(X)$ cannot have a greater number of any one factor than $a_1(X - \alpha_1)^{e_1-1} \dots (X - \alpha_m)^{e_m-1}$. Thus,

$g_i \leq e_i - 1$, $i = 1, \dots, v$ which implies that each factor of $K(X)$ is contained in the factorization of $D(X)$. Hence, $K(X)$ divides $D(X)$. Therefore, $D(X)$ is the monic greatest common divisor of $P(X)$ and $P'(X)$.

$D(X)$ is unique since if $\bar{D}(X)$ is a monic g.c.d. of $P(X)$ and $P'(X)$, then $\bar{D}(X)$ divides $P(X)$ and $P'(X)$ and hence $D(X)$. But $D(X)$ divides both $P(X)$ and $P'(X)$ and hence $\bar{D}(X)$. Thus, $D(X) = \bar{D}(X)$ since both are monic polynomials.

Consider the polynomial $H(X)$ obtained by dividing $P(X)$ by its monic g.c.d., $D(X)$.

$$\begin{aligned} H(X) &= P(X)/D(X) \\ &= a_1 \prod_{i=1}^m (X - \alpha_i)^{e_i} / \prod_{i=1}^m (X - \alpha_i)^{e_i-1} \\ &= a_1 \prod_{i=1}^m (X - \alpha_i). \end{aligned}$$

The zeros of $H(X)$ are all simple zeros and are also all the distinct zeros of $P(X)$. Use of the g.c.d. method involves computation of $H(X)$ when given $P(X)$.

In order to obtain $H(X)$, a computational algorithm is necessary to find the g.c.d. of $P(X)$ and $P'(X)$. The general method for computing the g.c.d. of two polynomials is as follows: Let $R_0(X)$ and $R_1(X)$ be two polynomials having degrees N_0 and N_1 respectively such that $N_1 \leq N_0$. The g.c.d. of $R_0(X)$ and $R_1(X)$ is desired. By the division algorithm, there exists polynomials $S_1(X)$ and $R_2(X)$ such that $R_0(X) = R_1(X) S_1(X) + R_2(X)$ where either $R_2(X) = 0$ or $\deg. R_2(X) < \deg. R_1(X)$. Similarly if $R_2(X) \neq 0$, there exists polynomials $S_2(X)$ and $R_3(X)$ such that $R_1(X) = S_2(X) R_2(X) + R_3(X)$ where either $R_3(X) = 0$ or $\deg. R_3(X) < \deg. R_2(X)$. Continuing in the

above manner, suppose $R_i(X)$ and $R_{i+1}(X)$ have been found where $\deg. R_{i+1}(X) < \deg. R_i(X)$. Then there exists polynomials $R_{i+2}(X)$ and $S_{i+1}(X)$ such that $R_i(X) = R_{i+1}(X) S_{i+1}(X) + R_{i+2}(X)$ where either $R_{i+2}(X) = 0$ or $\deg. R_{i+2}(X) < \deg. R_{i+1}(X)$. Then we obtain a sequence $R_0(X), R_1(X), \dots, R_K(X), R_{K+1}(X)$ such that $\deg. R_i(X) < \deg. R_{i-1}(X)$, $i = 1, 2, \dots, K+1$. Since a polynomial cannot have degree less than zero, the above process, in a finite number of steps (at most N_1), results in polynomials $R_{K-1}(X)$, $S_K(X)$ and $R_K(X)$ with $\deg. R_K(X) < \deg. R_{K-1}(X)$ such that $R_{K-1}(X) = R_K(X) S_K(X) + R_{K+1}(X)$ and $R_{K+1}(X) = 0$.

As an example, consider the problem of finding the g.c.d. of $R_1(X) = X^3 - 1$ and $R_0(X) = X^4 + X^3 + 2X^2 + 1$. Then

$$R_0(X) = R_1(X) (X + 1) + 2X^2 + 2X + 2$$

where $R_2(X) = 2X^2 + 2X + 2$. But

$$R_1(X) = R_2(X) \left(\frac{1}{2}X - \frac{1}{2}\right) + 0$$

where $R_3(X) = 0$. This theory leads to the following theorem.

Theorem 4.2. Let the sequence $R_0(X), R_1(X), \dots, R_K(X), R_{K+1}(X)$ be defined as above. Then $R_K(X)$ is the greatest common divisor of $R_0(X)$ and $R_1(X)$.

Proof. Since $R_{K-1}(X) = R_K(X) S_K(X) + R_{K+1}(X)$ where $R_{K+1}(X) = 0$, then $R_{K-1}(X) = R_K(X) S_K(X)$ and thus $R_K(X)$ divides $R_{K-1}(X)$. Also

$$\begin{aligned} R_{K-2}(X) &= R_{K-1}(X) S_{K-1}(X) + R_K(X) \\ &= R_K(X) S_K(X) S_{K-1}(X) + R_K(X) \end{aligned}$$

$$= R_K(X) [S_K(X) S_{K-1}(X) + 1].$$

Thus, $R_K(X)$ divides $R_{K-2}(X)$. Suppose $R_K(X)$ divides $R_i(X)$ for $0 \leq i < i+1 \leq K$. This produces

$$R_i(X) = R_K(X) F_i(X) \quad \text{and}$$

$$R_{i+1}(X) = R_K(X) F_{i+1}(X)$$

where $F_i(X)$ and $F_{i+1}(X)$ are polynomials. Then

$$\begin{aligned} R_{i-1}(X) &= R_i(X) S_i(X) + R_{i+1}(X) \\ &= [R_K(X) F_i(X)] S_i(X) + R_K(X) F_{i+1}(X) \\ &= R_K(X) [F_i(X) S_i(X) + F_{i+1}(X)] \\ &= R_K(X) Q(X). \end{aligned}$$

Therefore, $R_K(X)$ divides $R_{i-1}(X)$. Since the sequence $R_0(X), R_1(X), \dots, R_K(X), R_{K+1}(X)$ is finite, then by induction $R_K(X)$ divides both $R_0(X)$ and $R_1(X)$. Reversing the process, if $L(X)$ divides both $R_0(X)$ and $R_1(X)$, then

$$R_0(X) = L(X) W(X)$$

$$R_1(X) = L(X) Y(X)$$

for some polynomials $W(X)$ and $Y(X)$. But by the division algorithm

$$R_0(X) = R_1(X) S(X) + R_2(X).$$

Substitution yields

$$L(X) W(X) - L(X) Y(X) S(X) = R_2(X)$$

$$L(X) [W(X) - Y(X) S(X)] = R_2(X)$$

$$L(X) Z(X) = R_2(X).$$

Hence, $L(X)$ divides $R_2(X)$. Similarly if $L(X)$ divides $R_i(X)$ and $R_{i+1}(X)$, then $L(X)$ divides $R_{i+2}(X)$. Continuing inductively leads to the conclusion that $L(X)$ divides $R_K(X)$. Therefore, $R_K(X)$ divides $R_0(X)$ and $R_1(X)$ and any polynomial that divides $R_0(X)$ and $R_1(X)$ also divides $R_K(X)$. Thus, $R_K(X)$ is the greatest common divisor of $R_0(X)$ and $R_1(X)$.

The above theorem tells how to obtain the greatest common divisor of two polynomials. A machine orientated method is now developed for computing the sequence of $R_j(X)$'s. Beginning the sequence with $R_0(X)$ and $R_1(X)$, the polynomial $R_{i+1}(X)$ of the sequence is derived from $R_i(X)$ and $R_{i-1}(X)$ as follows: Let $R_{i-1}(X)$ of degree N_{i-1} be given by

$$\begin{aligned} R_{i-1}(X) \\ = r_{i-1,1} X^{N_{i-1}} + r_{i-1,2} X^{N_{i-1}-1} + \dots + r_{i-1,N_{i-1}} X + r_{i-1,N_{i-1}+1} \end{aligned}$$

and $R_i(X)$ of degree N_i be given by

$$R_i(X) = r_{i,1} X^{N_i} + r_{i,2} X^{N_i-1} + \dots + r_{i,N_i} X + r_{i,N_i+1}$$

where $N_i \leq N_{i-1}$. Define $U_1(X)$ by

$$U_1(X) = (r_{i-1,1} / r_{i,1}) X^{N_{i-1}-N_i}.$$

Then define $T_1(X)$ by $T_1(X) = R_{i-1}(X) - R_i(X) U_1(X)$ which can be expressed as

$$\begin{aligned} T_1(X) &= [r_{i-1,1} - r_{i,1} (r_{i-1,1} / r_{i,1})] X^{N_{i-1}} \\ &\quad + [r_{i-1,2} - r_{i,2} (r_{i-1,1} / r_{i,1})] X^{N_{i-1}-1} \\ &\quad + \dots \\ &\quad + [r_{i-1, N_{i-1}+1} - r_{i, N_{i-1}+1} (r_{i-1,1} / r_{i,1})] \end{aligned}$$

where $r_{i,j} = 0$ for $j > N_i+1$, or by

$$T_1(X) = t_{1,1} X^{M_1} + t_{1,2} X^{M_1-1} + \dots + t_{1,M_1} X + t_{1,M_1+1}$$

where $\deg. T_1(X) = M_1$. Three cases must now be considered.

- (1) If $T_1(X) = 0$, then $R_i(X) = R_K(X)$; that is, $R_i(X)$ is the g.c.d. of $R_0(X)$ and $R_1(X)$.
- (2) If $T_1(X) \neq 0$ and $\deg. T_1(X) < N_i$, then $R_{i+1}(X) = T_1(X)$.
- (3) If $T_1(X) \neq 0$ and $\deg. T_1(X) \geq N_i$, then define $U_2(X)$ by

$$U_2(X) = (t_{1,1} / r_{i,1}) X^{M_1 - N_i}.$$

Define $T_2(X) = T_1(X) - U_2(X) R_i(X)$ which can be expressed by

$$\begin{aligned} T_2(X) &= [t_{1,1} - (t_{1,1} / r_{i,1}) r_{i,1}] X^{M_1-1} \\ &\quad + [t_{1,2} - (t_{1,1} / r_{i,1}) r_{i,2}] X^{M_1-2} \end{aligned}$$

+ ...

$$+ [t_{1,M_1+1} - (t_{1,1}/r_{i,1}) r_{i,M_1+1}]$$

where $r_{i,j} = 0$ for $j > N_i+1$. Again three cases are considered.

- (1) If $T_2(X) = 0$, then $R_i(X)$ is the g.c.d. of $R_0(X)$ and $R_1(X)$.
- (2) If $T_2(X) \neq 0$ and $\deg. T_2(X) < \deg. R_i(X)$, then $R_{i+1}(X) = T_2(X)$.
- (3) If $T_2(X) \neq 0$ and $\deg. T_2(X) \geq N_i$, then similarly define $U_3(X)$ and $T_3(X)$ and repeat as above.

Since $\deg. T_{i+1}(X) < \deg. T_i(X)$, then this process is finite (not to exceed N_{i-1}) ending, for some integer S , in $T_S(X)$ such that

- (1) $T_S(X) = 0$ and $R_i(X)$ is the g.c.d. of $R_0(X)$ and $R_1(X)$ or
- (2) $T_S(X) \neq 0$ but $\deg. T_S(X) < \deg. R_i(X)$, in which case $T_S(X) = R_{i+1}(X)$.

Thus, using this algorithm and given $R_0(X)$ and $R_1(X)$, the sequence $R_0(X), R_1(X), R_2(X), \dots, R_i(X), R_{i+1}(X)$ can be generated such that either

- (1) $R_{i+1}(X) = 0$ and $R_i(X)$ is the g.c.d. of $R_0(X)$ and $R_1(X)$ or
- (2) $R_{i+1}(X) \neq 0$ and $N_{i+1} < N_i$. In a finite number of iterations, $R_k(X)$, the g.c.d. of $R_0(X)$ and $R_1(X)$, can be obtained.

Recall that the desire is to obtain the polynomial $H(X) = P(X)/D(X)$ where $D(X)$ is the g.c.d. of $P(X)$ and $P'(X)$. Thus, after obtaining $D(X)$

by the above algorithm, it is necessary to divide $P(X)$ by $D(X)$ obtaining $H(X)$ all whose zeros are simple.

Once $H(X)$ is obtained, an appropriate method such as Newton's method is applied to extract the zeros of $H(X)$. This gives all the zeros of $P(X)$.

As in Newton's or Muller's method, the zeros may be checked for accuracy and possibly improved by using them as initial approximations with the particular method applied to the full (undeflated) polynomial, $P(X)$.

2. Determining Multiplicities

After all zeros of $P(X)$ are found, the multiplicity of each is determined as follows: If $P(X)$ has m distinct zeros, then $P(X)$ can be written as

$$P(X) = a_1 (X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \dots (X - \alpha_m)^{e_m}, \quad (m \leq N)$$

where α_i is a zero of $P(X)$ and e_i is the multiplicity of α_i ($i = 1, 2, \dots, m$). Consider the zero α_j . Dividing out the term $(X - \alpha_j)$ by deflating $P(X)$ using synthetic division gives $P_1(X)$ of degree $N-1$ which can be written as

$$P_1(X) = (X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \dots (X - \alpha_j)^{e_j - 1} \dots (X - \alpha_m)^{e_m}.$$

Evaluating $P_1(X)$ at α_j gives $P_1(\alpha_j) = 0$ if $e_j > 1$. Thus, α_j is a zero of multiplicity at least equal to two. Then $P_1(X)$ is deflated obtaining $P_2(X)$ and $P_2(\alpha_j)$ is found. If $P_2(\alpha_j) = 0$, α_j is of

multiplicity at least three. This process is continued until a deflated polynomial $P_K(X)$ is found such that either $\deg. P_K(X) = 0$ or $P_K(\alpha_j) \neq 0$. Thus, if the value of the original polynomial, $P(X)$, and the next L successively deflated polynomials is zero at the root α_j and $P_{L+1}(\alpha_j) \neq 0$, then α_j is of multiplicity $L+1$.

The iteration formulas for deflation by synthetic division are derived in [6, PP. 106-107] and are given in Theorem 2.3 in Chapter II, § 1.

3. Procedure for the G.C.D. Method

The basic steps performed by the greatest common divisor method are listed sequentially as follows:

1. Given a polynomial, $P(X)$, in the form

$$P(X) = a_1 X^N + a_2 X^{N-1} + \dots + a_N X + a_{N+1}.$$

2. Calculate the derivative, $P'(X)$, of $P(X)$ in the form

$$P'(X) = b_1 X^{N-1} + b_2 X^{N-2} + \dots + b_N \text{ where } b_1 = N a_1, \\ b_2 = (N-1) a_2, \dots, b_N = a_N.$$

3. Find $D(X)$, the g.c.d. of $P(X)$ and $P'(X)$ using the algorithms developed above.
4. Calculate $H(X) = P(X)/D(X)$, the polynomial having only simple zeros.
5. Use some appropriate method to extract the zeros of $H(X)$.
6. Determine the multiplicity of each of the zeros obtained in step 5.

CHAPTER V

LEHMER'S METHOD

1. Derivation of the Algorithm

Lehmer's method in [4] is an iterative procedure designed to find numerical approximations to the zeros of polynomials with real or complex coefficients. The method employs a systematic search of the complex plane for the set of zeros which may be any arbitrary finite set of points. Lehmer's method has a slower convergence rate than methods with quadratic convergence but is well suited for high speed computers and is particularly useful as a subroutine by which polynomials created internally during the computer's work on a larger problem can be solved without external supervision since no initial approximation of a zero is required.

The heart of this search procedure lies in repeatedly applying and answering the basic question,

"Does a given polynomial have a zero inside a given circle?"

More precisely, the method can be divided into steps as follows. Let

$$P(X) = a_1 X^N + a_2 X^{N-1} + \dots + a_N X + a_{N+1} \quad (5-1)$$

be a polynomial with complex coefficients such that $a_1 \neq 0$ and $P(0) \neq 0$.

Step 1. This step begins by asking the question, "Does $P(X)$ have

a zero inside the unit circle?" If the answer is yes, then the radius is repeatedly halved and the basic question asked until an annulus

$$R < |Z| < 2R \quad (R \text{ is radius of inner circle})$$

is found containing a zero, X_j , of $P(X)$ while the inner circle is free of zeros. Note that $R = \frac{1}{2^K}$ for some integer K since the radius is repeatedly halved. If the answer to the question is no, the radius is repeatedly doubled until the above conditions are satisfied where, in this case, $R = 2^M$ for some integer M . This resulting annulus can be completely covered by 8 overlapping circles (Figure 2.1) each of radius $r_1 = (\frac{5}{6})R$ with centers located at the vertices of a regular octagon inscribed in a circle of radius $(\frac{5}{3})R$, that is, the points

$$\left(\frac{5}{3}\right)R e^{\frac{\pi K}{4} i}, \quad K = 0, 1, \dots, 7.$$

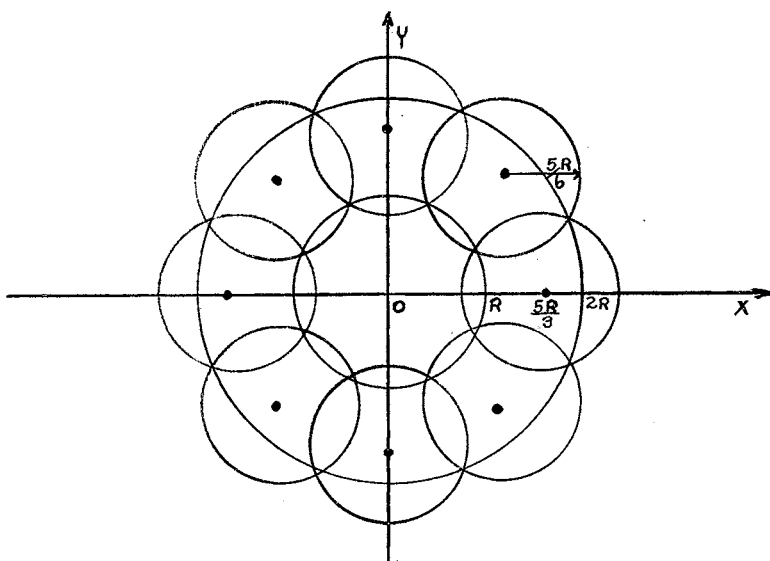


Figure 2.1. Covering an Annulus With Eight Overlapping Circles

Using the conversion $r e^{i\theta} = r (\cos \theta + i \sin \theta)$, the coordinates of these centers can be written in rectangular form as

$$\left(\frac{5}{3}\right)R \cos\left(\frac{\pi K}{4}\right) + i \left(\frac{5}{3}\right)R \sin\left(\frac{\pi K}{4}\right), \quad K = 0, 1, \dots, 7.$$

Starting with $K = 0$, the basic question is asked of each of these circles until one is found containing the root X_j . Denoting the center of this circle by C_1 completes step 1.

Step 2. Using C_1 as the origin, step 2 repeatedly halves the radius, R , until an annulus

$$R_1 < |Z - C_1| < 2R_1$$

is found containing a zero while the inner circle of radius R_1 is free of roots. Here

$$R_1 = \left(\frac{1}{2^K}\right)r_1 = \frac{1}{2^K} \left(\frac{5R}{6}\right)$$

where K is a positive integer. $|Z - C_1|$ is the distance of a point Z in the annulus from the center C_1 . Again this annulus can be covered by 8 overlapping circles each of radius $r_2 = \left(\frac{5}{6}\right)R_1$ with centers at the points

$$\frac{5R_1}{3} e^{\frac{\pi K}{4} i}, \quad K = 0, 1, \dots, 7.$$

Note that r_1 , r_2 , R_1 , and R are related by the following:

$$r_2 = \frac{5R_1}{6} = \frac{5}{6} \left(\frac{1}{2^K}\right)r_1 \leq \frac{5}{6} \left(\frac{r_1}{2}\right) = \frac{5r_1}{12} = \frac{5}{12} \left(\frac{5R}{6}\right) = \frac{25R}{72}. \quad (5-1.1)$$

Beginning with $K = 0$, repeated application of the basic question to each of these 8 circles determines one containing X_j . Due to overlapping, it is possible for two circles to contain X_j , in which case, the circle corresponding to the smaller value of K is used. Denoting the center of this circle by C_2 completes step 2.

After the completion of step M of this procedure, a circle with center C_M and radius r_M is found containing the zero X_j . From repeated application of the inequality (5-1.1), we obtain

$$r_M \leq 2\left(\frac{5}{12}\right)^M R.$$

Thus, by choosing M sufficiently large, a circle of sufficiently small radius is found containing the zero X_j . The coordinates of C_M with respect to the original origin, $(0,0)$, of the complex plane is then the desired approximation to the zero, X_j .

After the zero X_j has been determined within the required degree of accuracy, the multiplicity of X_j is determined as explained in Chapter II, § 5. $P(X)$ is then deflated by synthetic division according to Theorem 2.3 of Chapter II. This deflation removes X_j , as a point, from the complex plane, thus rendering X_j no longer available to be recaptured on any succeeding application of the method in extracting the remaining zeros. $P(X)$ is then replaced by $P_1(X)$ and the above procedure applied to extract another root. Therefore, by successive deflations, all the zeros of $P(X)$ can be approximated.

Figure 3 gives the geometric interpretation of Lehmer's method through step 3 of the above procedure.

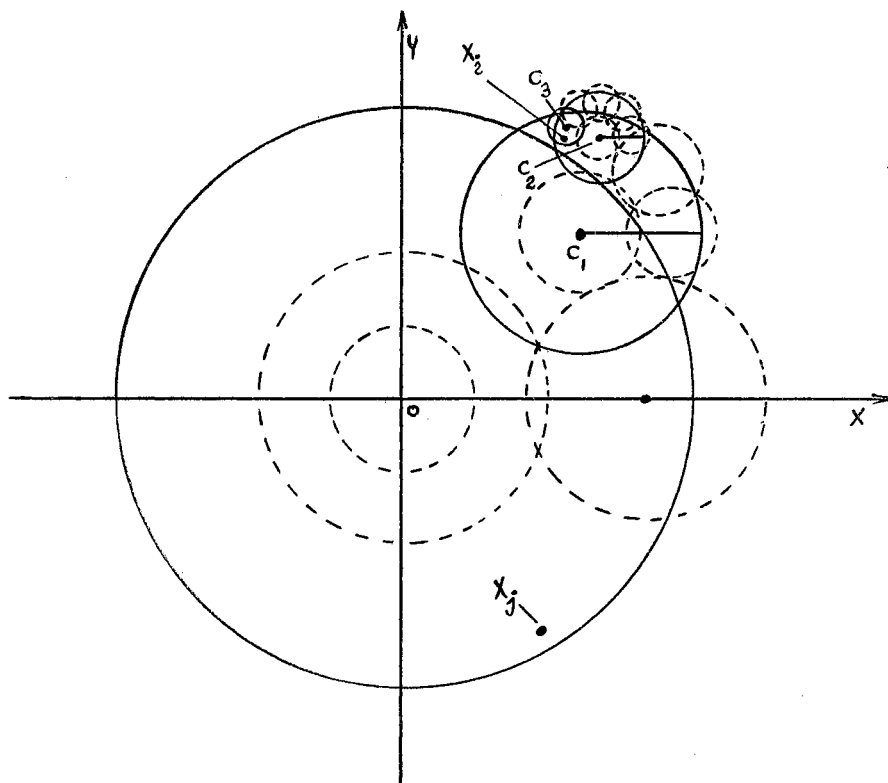


Figure 3. Geometric Interpretation of Lehmer's Method

The dotted circles indicate the absence of zeros within them. $C_1, C_2,$ and C_3 are the centers of the resulting circles after completion of steps 1, 2, and 3, respectively. The zero determining the first annulus is X_j but X_1 is captured by the process rather than X_j . Thus, if two zeros have approximately the same modulus, the one with the smaller argument is captured first. Lehmer's method extracts zeros roughly in the order of increasing modulus.

In order to apply the above procedure, a machine orientated method is needed to answer the basic question:

"Does $P(X)$ have a zero inside the circle with center C and radius ρ "; that is, the circle $|X - C| = \rho$?

By applying the transformation $Z = \rho Z$ which corresponds to a stretching of the complex plane if $\rho > 1$ or a shrinking of the complex plane if $\rho < 1$, and the translation $Z = Z + C$, the given circle, $|X - C| = \rho$ can be replaced by the unit circle, $|Z| = 1$, using the polynomial

$$G(Z) = P(\rho Z + C).$$

If X_j is a zero of $P(X)$ then $\beta = (X_j - C)/\rho$ is a zero of $G(X)$ since

$$\begin{aligned} G(\beta) &= P\left[\rho\left(\frac{X_j - C}{\rho}\right) + C\right] \\ &= P(X_j) \\ &= 0. \end{aligned}$$

Also $|\beta| < 1$ if and only if $|X_j - C| < \rho$; that is, β is a zero of $G(X)$ inside the unit circle if and only if X_j is a zero of $P(X)$ inside the circle $|X - C| = \rho$. Thus, the existence of the zero β implies the existence of the zero X_j and conversely. Thus, the basic question is equivalent to:

"Does $G(X)$ have a zero inside the unit circle Γ ?"

In order to answer this question, a sequence of polynomials is constructed from $G(X)$ as follows: Let $G(X)$ be given by

$$G(X) = b_1 X^N + b_2 X^{N-1} + \dots + b_N X + b_{N+1} \quad (5-2)$$

where b_1 and $b_{N+1} \neq 0$ and the coefficients are complex. Note that $G(X)$

is constructed from $P(X)$ by the linear transformation $X = \rho X + c$ and hence $\deg. G(X) = N = \deg. P(X)$. Denoting the conjugate of $b_j = u + iv$ by $\bar{b}_j = u - iv$, let $G^*(X)$ be defined by

$$G^*(X) = \bar{b}_{N+1}X^N + \bar{b}_N X^{N-1} + \dots + \bar{b}_2 X + \bar{b}_1;$$

that is, the coefficients of $G^*(X)$ are the conjugates of those of $G(X)$ but in reverse order. Define the transformation T on $G(X)$ by

$$T(G(X)) = \bar{b}_{N+1}G(X) = b_1 G^*(X).$$

$T(G(X))$ is a polynomial of degree less than $\deg. G(X)$ since

$$\begin{aligned} T(G(X)) &= \bar{b}_{N+1}(b_1 X^N + b_2 X^{N-1} + \dots + b_N X + b_{N+1}) \\ &\quad - b_1(\bar{b}_{N+1} X^N + \bar{b}_N X^{N-1} + \dots + \bar{b}_2 X + \bar{b}_1) \\ &= \bar{b}_{N+1} b_1 X^N + \bar{b}_{N+1} b_2 X^{N-1} + \dots + \bar{b}_{N+1} b_N X + \bar{b}_{N+1} b_{N+1} \\ &\quad - b_1 \bar{b}_{N+1} X^N - b_1 \bar{b}_N X^{N-1} - \dots - b_1 \bar{b}_2 X - b_1 \bar{b}_1 \\ &= (\bar{b}_{N+1} b_1 - b_1 \bar{b}_{N+1}) X^N + (\bar{b}_{N+1} b_2 - b_1 \bar{b}_N) X^{N-1} \\ &\quad + \dots + (\bar{b}_{N+1} b_N - b_1 \bar{b}_2) X + (\bar{b}_{N+1} b_{N+1} - b_1 \bar{b}_1) \\ &= t_1 X^{N-1} + t_2 X^{N-2} + \dots + t_{N-1} X + t_N \end{aligned}$$

where

$$t_i = \bar{b}_{N+1} b_{i+1} - b_1 \bar{b}_{N-i+1}, \quad i = 1, 2, \dots, N.$$

The constant term of $T(G(X))$ is

$$t_N = \bar{b}_{N+1} b_{N+1} - b_1 \bar{b}_1 = |b_{N+1}|^2 - |b_1|^2 \quad (5-3)$$

and is, therefore, real because the product of a complex number and its conjugate is real. In fact,

$$\begin{aligned} T(G(0)) &= \bar{b}_{N+1} b_{N+1} - b_1 \bar{b}_1 \\ &= |b_{N+1}|^2 - |b_1|^2. \end{aligned} \quad (5-4)$$

If $T(G(0)) \neq 0$, applying this transformation, T , to $T(G(X))$ gives $T^2(G(X)) = T[T(G(X))]$ where $\deg. T^2(G(X)) < \deg. T(G(X))$. Continuing, we obtain a sequence of polynomials

$$T(G(X)), T^2(G(X)), T^3(G(X)), \dots, T^K(G(X)) \quad (5-5)$$

where

$$T^i(G(X)) = T[T^{i-1}(G(X))], \quad i = 2, \dots, K.$$

Let $\deg. T^i(G(X)) = d_i$. Then $N > d_1 > d_2 > \dots > d_K \geq 0$ since, from above, it was shown that T reduces the degree of the polynomial by at least one.

The polynomial $T^{i+1}(G(X))$ is obtained from $T^i(G(X))$ as follows.

Let $T^i(G(X))$ be denoted by

$$T^i(G(X)) = r_1 X^{d_i} + r_2 X^{d_i-1} + \dots + r_{d_i} X + r_{d_i+1}, \quad r_{d_i+1} \neq 0.$$

Then

$$[T^i(G(X))]^* = \bar{r}_{d_i+1} X^{d_i} + \bar{r}_{d_i} X^{d_i-1} + \dots + \bar{r}_2 X + \bar{r}_1.$$

T applied to $T^i(G(X))$ gives

$$\begin{aligned} T^{i+1}(G(X)) &= T[T^i(G(X))] \\ &= \bar{r}_{d_i+1} T^i(G(X)) - r_1 [T^i(G(X))]^* \\ &= \sum_{j=1}^{d_i} \left(\bar{r}_{d_i+1} r_{j+1} - r_1 \bar{r}_{d_i-j+1} \right) X^{d_i-j}. \end{aligned}$$

Since $\deg. P(X) = N$ is finite, then in a finite number of steps (at most $N+1$) a polynomial $T^K(G(X))$ is obtained such that $T^K(G(0)) = 0$.

This is due to the following reasoning. $T^i(G(0))$ is the constant, real term of $T^i(G(X))$. If $T^i(G(0)) = 0$, we are done. If $T^i(G(0)) \neq 0$, then in at most N steps, we obtain a polynomial $T^{K-1}(G(X))$ of degree 0; that is, $T^{K-1}(G(X)) = c$ (constant). Then $T^K(G(X)) = \bar{c}c - c\bar{c} = 0$. Hence $T^K(G(0)) = 0$.

Finally, the question, "Does the polynomial $G(X)$ have a zero inside the unit circle Γ ?" can be answered by the following two theorems.

Theorem 5.2. Let $G(X)$ have no roots on the unit circle Γ . Suppose $G(0) \neq 0$. If for some $h > 0$, $T^h(G(0)) < 0$, then $G(X)$ has at least one zero inside Γ . If, instead, $T^i(G(0)) > 0$ for $1 \leq i < K$ and $T^{K-1}(G(X))$ is a constant, then no roots of $G(X)$ lie inside Γ .

The proof of this theorem can be found in [4, PP. 154-156]. The Cauchy Integral Formula, Rouché's Theorem, and the Argument Principle

from complex analysis are used in the proof of Theorem 5.2. Statements and proofs of these theorems can be found in [8, PP. 293-295], [9, P. 145], and [9, PP. 142-143], respectively.

Theorem 5.3. Theorem 5.2 remains true if we weaken its hypothesis by deleting its first sentence. Let $G(X)$ be a polynomial with $G(0) \neq 0$. If for some $h > 0$, $T^h(G(0)) < 0$, then $G(X)$ has at least one zero inside the unit circle Γ . If, instead, $T^i(G(0)) > 0$ for $1 \leq i < K$ and $T^{K-1}(G(X))$ is a constant, then $G(X)$ has no roots inside Γ .

This theorem is proved in [4, PP. 156-157].

It may accidentally happen in certain specially made polynomials with integer coefficients that $T^{K-1}(G(X))$ of Theorem 5.2 is not a constant. For example, consider the polynomial

$$G(X) = 6X^4 - 35X^3 + 62X^2 - 35X + 6.$$

In this case $G^*(X) = G(X)$ so that $T(G(X))$ vanishes identically. When this happens, it is better to ask the basic question on a concentric circle of radius larger than the present one by a factor such as $\frac{3}{2}$. This also is good practice in case the value $T^i(G(0))$ is so small in magnitude that the sign is uncertain due to roundoff error, or when $G(0) = 0$.

To illustrate this algorithm, consider the following example. Let $G(X)$ be the polynomial

$$G(X) = (1 + i)X^2 + 2X + (2 + 2i).$$

Then

$$G^*(X) = (2 - 2i)X^2 + 2X + (1 - i)$$

$$\begin{aligned} T(G(X)) &= (2 - 2i)G(X) - (1 + i)G^*(X) \\ &= (2 - 2i)[(1 + i)X^2 + 2X + 2 + 2i] \\ &\quad - (1 + i)[(2 - 2i)X^2 + 2X + (1 - i)] \\ &= [(2 - 2i)(1 + i) - (1 + i)(2 - 2i)]X^2 \\ &\quad + [(2 - 2i)2 - (1 + i)2]X \\ &\quad + [(2 - 2i)(2 + 2i) - (1 + i)(1 - i)] \\ &= (2 - 6i)X + 6 \end{aligned}$$

Thus, $T(G(0)) = 6$

$$T^*(G(X)) = 6X + (2 + 6i)$$

$$\begin{aligned} T^2(G(X)) &= T[T(G(X))] \\ &= 6T(G(X)) - (2 - 6i)T^*(G(X)) \\ &= 6[(2 - 6i)X + 6] - (2 - 6i)[6X + (2 + 6i)] \\ &= [6(2 - 6i) - (2 - 6i)6]X + 36 - (2 - 6i)(2 + 6i) \\ &= 36 - (4 + 36) \\ &= 36 - 40 \\ &= -4 \end{aligned}$$

Thus, $T^2(G(0)) = -4$.

Therefore, by Theorem 5.3, $G(X)$ has at least one root inside Γ .

2. Convergence of Lehmer's Method

Convergence of this searching technique is somewhat slower than other methods but is somewhat that of the geometric progression of ratio $\frac{5}{12}$. It has been shown that after completion of step K , a circle of radius less than $2R\left(\frac{5}{12}\right)^K$ is obtained containing a root. Twenty-seven steps give a radius of length less than $R \cdot 10^{-10}$. Hence, for K large enough, the center of the circle is a good approximation to the zero. If the value of $P(X)$ at the center is sufficiently near zero, then convergence is obtained.

3. Determining Multiplicities

Multiplicities of the zeros are determined as described in Chapter II, § 5.

4. Procedure for Lehmer's Method

The basic steps performed by Lehmer's method are listed sequentially as follows:

1. Use Theorem 5.3 to determine if $P(X)$ has a zero inside the unit circle with center at the origin.
 - a. If the answer is yes, halve the radius until a circle of radius R is found such that $P(X)$ has a zero inside the circle $|X| = 2R$ but none inside $|X| = R$.
 - b. If the answer is no, double the radius until such an R is found.

2. Beginning with $K = 0$, translate the origin to the point $c = \left(\frac{5R}{3}\right) \left(\cos\left(\frac{\pi K}{4}\right) + i \sin\left(\frac{\pi K}{4}\right)\right)$ using the radius $\rho = \left(\frac{5}{6}\right)R$. Call the resulting polynomial $G(X)$.
3. Using Theorem 5.3 determine if $G(X)$ has a zero inside this circle.
 - a. If it does, go to step 4.
 - b. If it does not, increase K by 1 and go to step 2 with R replaced by ρ .
 - c. If Theorem 5.3 cannot be applied, replace R by $\left(\frac{3}{2}\right)R$ and return to step 2.
4. Save this center as the next approximation to the zero and halve the radius until a circle of radius R_1 is found such that $G(X)$ has a zero inside $|X - c| = 2R_1$ but none in $|X - c| = R$.
5. Test for convergence. Is $P(c)$ sufficiently near zero to be called a root? If it is not, return to step 2. If convergence is obtained, then
 - a. Save latest center as the approximation to the zero.
 - b. Deflate $P(X)$ using $(X - c)$, resulting in $P_1(X)$.
 - c. Replace $P(X)$ by $P_1(X)$.
 - d. Return to step 1.

Given the polynomial $F(X) = f_1 X^M + f_2 X^{M-1} + \dots + f_M X + f_{M+1}$, the sequence of steps to determine if $F(X)$ has a zero inside a circle of radius ρ and center c is listed in the order performed below.

Theorem 5.3 consists of steps 2 - 8.

1. Calculate $G(X) = F(\rho X + c)$ where ρ is the radius of the circle and c is its center.

$G(X) = g_1 X^S + g_2 X^{S-1} + \dots + g_S X + g_{S+1}$. This replaces the given circle with the unit circle Γ . Theorem 5.3 is applied to this $G(X)$. Set $j = 1$.

2. If $G(0) = 0$, replace ρ by $(\frac{3}{2})\rho$ and return to step 1.
If $G(0) \neq 0$, proceed to step 3.
3. Calculate $G^*(X) = \overline{g_{S+1}} X^S + \overline{g_S} X^{S-1} + \dots + \overline{g_2} X + \overline{g_1}$.
4. Calculate $T^j(G(X)) = \overline{g_{S+1}} G(X) - g_1 G^*(X)$.
5. Find $T^j(G(0))$.
 - a. If $T^j(G(0))$ is too small to determine sign, replace ρ by $(\frac{3}{2})\rho$ and return to step 1.
 - b. If $T^j(G(0)) < 0$, go to step 6.
 - c. If $T^j(G(0)) = 0$, go to step 7.
 - d. If $T^j(G(0)) > 0$, go to step 8.
6. $G(X)$ has a root inside Γ .
7. If $T^{j-1}(G(X))$ is a constant, $G(X)$ has no root inside Γ .
If $T^{j-1}(G(X))$ is not a constant, replace ρ by $(\frac{3}{2})\rho$ and return to step 1.
8. Replace $G(X)$ by $T^j(G(X))$, increase j by 1 and return to step 2.

5. Conclusion of the Method

In applying the transformation T , the coefficients of $T^K(G(X))$ may become very large or very small in absolute value. For example, the constant term of $T(G(X))$ is computed by equation (5-3). If the magnitude of the constant term of $G(X)$ is large compared to that of the leading coefficient, then repeated application of the transformation results in the constant term of $T^{i+1}(G(X))$ of sequence (5-5) having magnitude approximately double that of the constant term of $T^i(G(X))$. This problem was encountered when the program was run on the IBM S/360 which permits magnitudes between 10^{-77} and 10^{77} . The program could not be run on the IBM S/360 due to the limitation on the size of the exponent permitted.

CHAPTER VI

QUOTIENT-DIFFERENCE METHOD

1. Derivation of the Algorithm

The quotient-difference (Q-D) method in [3, PP. 162-179] is an iterative procedure producing simultaneous approximations to all zeros of a polynomial with real or complex coefficients. This method has the particular advantage that no initial approximations to the zeros are required, since no information is needed other than the degree of the polynomial and its coefficients. Due to roundoff error and slow convergence, the Q-D method is recommended only for the purpose of giving initial approximations to the zeros. These initial approximations are then used with another method, such as Newton's method, for obtaining final results.

Let $P(X)$ be a polynomial of degree N given by

$$P(X) = a_1 X^N + a_2 X^{N-1} + \dots + a_N X + a_{N+1} \quad (6-1)$$

where $a_1 \neq 0 \neq a_{N+1}$ and the coefficients are complex numbers. The quotient-difference algorithm consists of constructing, row after row, a two dimensional array of numbers called the Q-D scheme arranged as indicated below for the case of $N = 4$. The notation here is chosen to eliminate zero subscripts and does not agree with the notation in [3], [10], or [11].

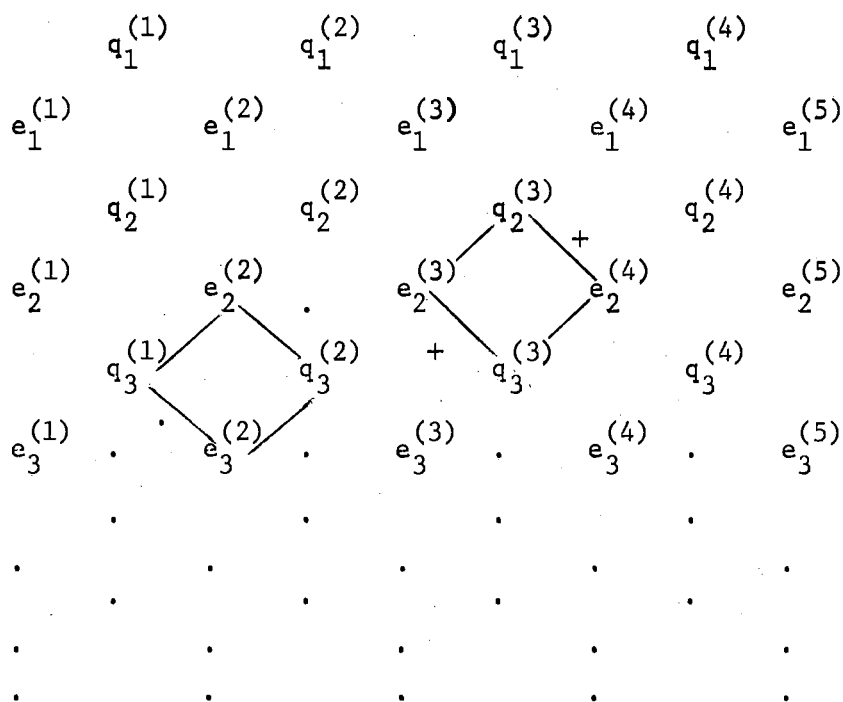


Figure 3.1. The Q-D Scheme

The entries in the array are related by the rhombus rules:

$$q_{n+1}^{(K)} = (e_n^{(K+1)} - e_n^{(K)}) + q_n^{(K)} \quad (K = 1, 2, \dots, N) \quad (6-2)$$

$$e_{n+1}^{(K)} = \frac{q_{n+1}^{(K)}}{q_{n+1}^{(K-1)}} e_n^{(K)} \quad (K = 2, 3, \dots, N+1).$$

where $n = 1, 2, 3, \dots$. As shown in the above diagram, these rules can be remembered as follows:

1. In a rhombus configuration with an e-element on top, the product of the northeast pair is equal to the product of the southwest pair.

2. In a rhombus configuration with a q -element on top, the sum of the northeast pair is equal to the sum of the southwest pair.

It is helpful to observe the following facts concerning the above diagram:

1. There are $2N+1$ columns.
2. The superscripts in each column are constant.
3. The subscripts in each row are constant.

Since each of the rhombus rules involves four adjacent elements, namely the vertices of a rhombus in the array, then the first two rows, the first column, and the last column must be known before the rules can be applied to generate, row after row, the remainder of the array. The first two rows are determined from the coefficients of the polynomial as follows:

$$q_1^{(1)} = -a_2/a_1$$

$$q_1^{(K)} = 0 \quad (K = 2, 3, \dots, N)$$

$$e_1^{(K)} = a_{K+1}/a_K \quad (K = 2, 3, \dots, N).$$

Here it is assumed that the coefficients are non-zero. The case in which some coefficients are zero will be treated later. The first and last column consists entirely of zeros:

$$e_n^{(1)} = e_n^{(N+1)} = 0$$

The array now appears as follows for

$$P(X) = a_1 X^4 + a_2 X^3 + a_3 X^2 + a_4 X + a_5.$$

$e_n^{(1)}$	$q_n^{(1)}$	$e_n^{(2)}$	$q_n^{(2)}$	$e_n^{(3)}$	$q_n^{(3)}$	$e_n^{(4)}$	$q_n^{(4)}$	$e_n^{(5)}$
	$-\frac{a_2}{a_1}$		0		0		0	
0		$\frac{a_3}{a_2}$		$\frac{a_4}{a_3}$		$\frac{a_5}{a_4}$		0
	$q_2^{(1)}$		$q_2^{(2)}$		$q_2^{(3)}$		$q_2^{(4)}$	
0		$e_2^{(2)}$		$e_2^{(3)}$		$e_2^{(4)}$		0
	$q_3^{(1)}$		$q_3^{(2)}$		$q_3^{(3)}$		$q_3^{(4)}$	
0	.	$e_3^{(2)}$.	$e_3^{(3)}$.	$e_3^{(4)}$.	0
.
.
.

Figure 3.2. Starting the Q-D Scheme

The rhombus rules can be obtained as in [11, PP. 36-38]. This is accomplished as follows, keeping in mind that the notation is different from that in [11]. Let $P(X)$ be a polynomial of degree N given by equation (6-1). Assume for the present that the zeros X_1, X_2, \dots, X_N

are all simple and satisfy $|x_1| > |x_2| > \dots > |x_N| > 0$. Assume further that the coefficients are non-zero. $P(X)$ can be expressed in the form

$$P(X) = a_1 \prod_{i=1}^N (X - x_i)$$

or

$$\frac{1}{P(X)} = \frac{1}{a_1 \prod_{i=1}^N (X - x_i)}$$

Then $\frac{1}{P(X)}$ can be expressed as a sum of partial fractions:

$$\frac{1}{P(X)} = \frac{A_1}{X - x_1} + \frac{A_2}{X - x_2} + \dots + \frac{A_N}{X - x_N} \quad (6-3)$$

To verify that this is true, it is only necessary to show that there exists unique A_1, A_2, \dots, A_N such that equation (6-3) holds. This is done as follows. Combining terms in (6-3) results in

$$\frac{1}{P(X)} = \frac{A_1 \prod_{i=2}^N (X - x_i) + \dots + A_j \prod_{\substack{i=1 \\ i \neq j}}^N (X - x_i) + \dots + A_N \prod_{i=1}^{N-1} (X - x_i)}{\prod_{i=1}^N (X - x_i)} \quad (6-4)$$

Substituting into the left hand side of equation (6-4) gives

$$\frac{1}{a_1 \prod_{i=1}^N (X - X_i)} = \frac{\sum_{j=1}^N \left(A_j \prod_{\substack{i=1 \\ i \neq j}}^N (X - X_i) \right)}{\prod_{i=1}^N (X - X_i)}.$$

$$\frac{1}{a_1} = \sum_{j=1}^N \left(A_j \prod_{\substack{i=1 \\ i \neq j}}^N (X - X_i) \right).$$

Consider $\frac{1}{a_1}$ as a polynomial of degree $N-1$ all of whose coefficients are zero except the constant coefficient $\frac{1}{a_1}$. The right hand side is also a polynomial of degree $N-1$ since each of the N terms is the product of $N-1$ linear factors. Equating coefficients of these two polynomials gives N simultaneous equations in the N unknowns A_1, A_2, \dots, A_N . Solution of this system yields the result. The terms on the right hand side of equation (6-3) can be expanded in a geometric series.

$$\begin{aligned} \frac{A_r}{X - X_r} &= \frac{A_r}{X} \left(\frac{1}{1 - \frac{X_r}{X}} \right) \\ &= A_r \left(\frac{1}{X} + \frac{X_r}{X^2} + \frac{X_r^2}{X^3} + \dots \right). \end{aligned}$$

for all X such that $X > X_r$, since the sum of the geometric series

$$\frac{1}{X} + \frac{X_r}{X^2} + \frac{X_r^2}{X^3} + \dots$$

is

$$\frac{\frac{1}{X}}{1 - \frac{X_r}{X}} = \frac{1}{X} \left(\frac{1}{1 - \frac{X_r}{X}} \right), \quad \left| \frac{X_r}{X} \right| < 1.$$

Thus, if $X > X_1$, then $X > X_r$ for $r = 1, 2, \dots, N$, and

$$\begin{aligned} \frac{1}{P(X)} &= A_1 \left(\frac{1}{X} + \frac{X_1}{X^2} + \frac{X_1^2}{X^3} + \dots \right) \\ &+ A_2 \left(\frac{1}{X} + \frac{X_2}{X^2} + \frac{X_2^2}{X^3} + \dots \right) \\ &+ \dots \\ &+ A_N \left(\frac{1}{X} + \frac{X_N}{X^2} + \frac{X_N^2}{X^3} + \dots \right) \\ &= \sum_{j=1}^N A_j \left(\frac{1}{X} \right) + \sum_{j=1}^N A_j \left(\frac{X_j}{X^2} \right) + \sum_{j=1}^N A_j \left(\frac{X_j^2}{X^3} \right) + \dots \\ &= \sum_{i=0}^{\infty} \left[\sum_{j=1}^N A_j \left(\frac{X_j^i}{X^{i+1}} \right) \right]. \end{aligned}$$

Let

$$\alpha_i = \sum_{j=1}^N A_j X_j^i.$$

Then

$$\frac{1}{P(X)} = \sum_{i=0}^{\infty} \alpha_i \frac{1}{X^{i+1}}.$$

Computing the quotient between two consecutive coefficients gives

$$q_i^{(1)} = \frac{\alpha_{i+1}}{\alpha_i} = \frac{A_1 X_1^{i+1} + A_2 X_2^{i+1} + A_3 X_3^{i+1} + \dots + A_N X_N^{i+1}}{A_1 X_1^i + A_2 X_2^i + A_3 X_3^i + \dots + A_N X_N^i}.$$

Multiplying both numerator and denominator by $\frac{1}{A_1 X_1^{i+1}}$ gives

$$\begin{aligned} q_i^{(1)} &= \frac{A_1 X_1^{i+1} \left(\frac{1}{A_1 X_1^{i+1}} \right) + A_2 X_2^{i+1} \left(\frac{1}{A_1 X_1^{i+1}} \right) + \dots + A_N X_N^{i+1} \left(\frac{1}{A_1 X_1^{i+1}} \right)}{A_1 X_1^i \left(\frac{1}{A_1 X_1^{i+1}} \right) + A_2 X_2^i \left(\frac{1}{A_1 X_1^{i+1}} \right) + \dots + A_N X_N^i \left(\frac{1}{A_1 X_1^{i+1}} \right)} \\ &= \frac{1 + \frac{A_2}{A_1} \left(\frac{X_2}{X_1} \right)^{i+1} + \dots + \frac{A_N}{A_1} \left(\frac{X_N}{X_1} \right)^{i+1}}{\frac{1}{X_1} \left[1 + \frac{A_2}{A_1} \left(\frac{X_2}{X_1} \right)^i + \dots + \frac{A_N}{A_1} \left(\frac{X_N}{X_1} \right)^i \right]}. \end{aligned}$$

But $|X_K/X_1| < 1$ for $K = 2, 3, \dots, N$. Thus, as $i \rightarrow \infty$ the numerator approaches 1 while the denominator approaches $\frac{1}{X_1}$. Thus, we have

$$\lim_{i \rightarrow \infty} q_i^{(1)} = \frac{1}{\left(\frac{1}{X_1} \right)} = X_1.$$

Then

$$\lim_{i \rightarrow \infty} \frac{X_1 - q_i^{(1)}}{\left(\frac{X_2}{X_1} \right)^i} = \lim_{i \rightarrow \infty} X_1 \left[\frac{1 - \frac{1 + \frac{A_2}{A_1} \left(\frac{X_2}{X_1} \right)^{i+1} + \dots + \frac{A_N}{A_1} \left(\frac{X_N}{X_1} \right)^{i+1}}{1 + \frac{A_2}{A_1} \left(\frac{X_2}{X_1} \right)^i + \dots + \frac{A_N}{A_1} \left(\frac{X_N}{X_1} \right)^i}}{\left(\frac{X_2}{X_1} \right)^i} \right].$$

By combining terms in the numerator, combining like terms, and factoring, the right hand side of the preceding equation becomes

$$\begin{aligned} & \left[\frac{1 + \frac{A_2}{A_1} \left(\frac{X_2}{X_1}\right)^i + \dots + \frac{A_N}{A_1} \left(\frac{X_N}{X_1}\right)^i - 1 - \frac{A_2}{A_1} \left(\frac{X_2}{X_1}\right)^{i+1} - \dots - \frac{A_N}{A_1} \left(\frac{X_N}{X_1}\right)^{i+1}}{1 + \frac{A_2}{A_1} \left(\frac{X_2}{X_1}\right)^i + \dots + \frac{A_N}{A_1} \left(\frac{X_N}{X_1}\right)^i} \right] \\ \text{limit}_{i \rightarrow \infty} X_1 & \left[\frac{\left(\frac{X_2}{X_1}\right)^i}{\left(\frac{X_2}{X_1}\right)^i} \right] \\ & = \text{limit}_{i \rightarrow \infty} X_1 \left[\frac{\frac{A_2}{A_1} \left(\frac{X_2}{X_1}\right)^i \left(1 - \frac{X_2}{X_1}\right) + \frac{A_3}{A_1} \left(\frac{X_3}{X_1}\right)^i \left(1 - \frac{X_3}{X_1}\right) + \dots + \frac{A_N}{A_1} \left(\frac{X_N}{X_1}\right)^i \left(1 - \frac{X_N}{X_1}\right)}{\left(\frac{X_2}{X_1}\right)^i \left(1 + \frac{A_2}{A_1} \left(\frac{X_2}{X_1}\right)^i + \dots + \frac{A_N}{A_1} \left(\frac{X_N}{X_1}\right)^i\right)} \right] \\ & = \text{limit}_{i \rightarrow \infty} X_1 \left[\frac{\frac{A_2}{A_1} \left(1 - \frac{X_2}{X_1}\right) + \frac{A_3}{A_1} \left(\frac{X_3}{X_2}\right)^i \left(1 - \frac{X_3}{X_1}\right) + \dots + \frac{A_N}{A_1} \left(\frac{X_N}{X_2}\right)^i \left(1 - \frac{X_N}{X_1}\right)}{1 + \frac{A_2}{A_1} \left(\frac{X_2}{X_1}\right)^i + \dots + \frac{A_N}{A_1} \left(\frac{X_N}{X_1}\right)^i} \right] \end{aligned}$$

But $|X_K/X_2| < 1$ for $K = 3, 4, \dots, N$ and $|X_K/X_1| < 1$ for $K = 2, 3, \dots, N$.

Thus

$$\text{limit}_{i \rightarrow \infty} \frac{X_1 - q_i^{(1)}}{\left(\frac{X_2}{X_1}\right)^i} = X_1 \left[\frac{A_2}{A_1} \left(1 - \frac{X_2}{X_1}\right) \right]. \quad (6-5)$$

Similarly, replacing i by $i+1$:

$$\lim_{i \rightarrow \infty} \frac{x_1 - q_{i+1}^{(1)}}{\left(\frac{x_2}{x_1}\right)^i}$$

$$= \lim_{i \rightarrow \infty} x_1 \left[\frac{1 - \left(\frac{1 + \frac{A_2}{A_1} \left(\frac{x_2}{x_1}\right)^{i+2} + \dots + \frac{A_N}{A_1} \left(\frac{x_N}{x_1}\right)^{i+2}}{1 + \frac{A_2}{A_1} \left(\frac{x_2}{x_1}\right)^{i+1} + \dots + \frac{A_N}{A_1} \left(\frac{x_N}{x_1}\right)^{i+1}} \right)}{\left(\frac{x_2}{x_1}\right)^i} \right]$$

$$= \lim_{i \rightarrow \infty} x_1 \left[\frac{1 + \frac{A_2}{A_1} \left(\frac{x_2}{x_1}\right)^{i+1} + \dots + \frac{A_N}{A_1} \left(\frac{x_N}{x_1}\right)^{i+1} - 1 - \frac{A_2}{A_1} \left(\frac{x_2}{x_1}\right)^{i+2} - \dots - \frac{A_N}{A_1} \left(\frac{x_N}{x_1}\right)^{i+2}}{\left(\frac{x_2}{x_1}\right)^i \left(1 + \frac{A_2}{A_1} \left(\frac{x_2}{x_1}\right)^{i+1} + \dots + \frac{A_N}{A_1} \left(\frac{x_N}{x_1}\right)^{i+1} \right)} \right]$$

$$= \lim_{i \rightarrow \infty} x_1 \left[\frac{\frac{A_2}{A_1} \left(\frac{x_2}{x_1}\right)^{i+1} \left(1 - \frac{x_2}{x_1}\right) + \frac{A_3}{A_1} \left(\frac{x_3}{x_1}\right)^{i+1} \left(1 - \frac{x_3}{x_1}\right) + \dots + \frac{A_N}{A_1} \left(\frac{x_N}{x_1}\right)^{i+1} \left(1 - \frac{x_N}{x_1}\right)}{\left(\frac{x_2}{x_1}\right)^i \left(1 + \frac{A_2}{A_1} \left(\frac{x_2}{x_1}\right)^{i+1} + \dots + \frac{A_N}{A_1} \left(\frac{x_N}{x_1}\right)^{i+1} \right)} \right]$$

$$= \lim_{i \rightarrow \infty} x_1 \left[\frac{\frac{A_2}{A_1} \left(\frac{x_2}{x_1}\right) \left(1 - \frac{x_2}{x_1}\right) + \frac{A_3}{A_1} \left(\frac{x_3}{x_2}\right)^{i+1} \left(1 - \frac{x_3}{x_1}\right) + \dots + \frac{A_N}{A_1} \left(\frac{x_N}{x_2}\right)^{i+1} \left(1 - \frac{x_N}{x_1}\right)}{1 + \frac{A_2}{A_1} \left(\frac{x_2}{x_1}\right)^{i+1} + \dots + \frac{A_N}{A_1} \left(\frac{x_N}{x_1}\right)^{i+1}} \right]$$

Thus

$$\begin{aligned} \lim_{i \rightarrow \infty} \frac{x_1 - q_{i+1}^{(1)}}{\left(\frac{x_2}{x_1}\right)^i} &= x_1 \left(\frac{A_2}{A_1}\right) \left(\frac{x_2}{x_1}\right) \left(1 - \frac{x_2}{x_1}\right) \\ &= x_2 \left(\frac{A_2}{A_1}\right) \left(1 - \frac{x_2}{x_1}\right). \end{aligned} \quad (6-6)$$

Subtracting equation (6-6) from equation (6-5) yields

$$\lim_{i \rightarrow \infty} \frac{q_{i+1}^{(1)} - q_i^{(1)}}{\left(\frac{x_2}{x_1}\right)^i} = (x_1 - x_2) \left(\frac{A_2}{A_1}\right) \left(1 - \frac{x_2}{x_1}\right).$$

Similarly, as in deriving equations (6-5) and (6-6),

$$\lim_{i \rightarrow \infty} \frac{x_1 - q_{i+2}^{(1)}}{\left(\frac{x_2}{x_1}\right)^i} = \frac{x_2^2}{x_1} \left(\frac{A_2}{A_1}\right) \left(1 - \frac{x_2}{x_1}\right).$$

Then

$$\lim_{i \rightarrow \infty} \frac{q_{i+2}^{(1)} - q_{i+1}^{(1)}}{\left(\frac{x_2}{x_1}\right)^i} = \left(x_2 - \frac{x_2^2}{x_1}\right) \frac{A_2}{A_1} \left(1 - \frac{x_2}{x_1}\right).$$

Now let

$$e_i^{(2)} = q_{i+1}^{(1)} - q_i^{(1)}.$$

Then

$$\begin{aligned}
\lim_{i \rightarrow \infty} \frac{e_{i+1}^{(2)}}{e_i^{(2)}} &= \lim_{i \rightarrow \infty} \frac{q_{i+2}^{(1)} - q_{i+1}^{(1)}}{q_{i+1}^{(1)} - q_i^{(1)}} \\
&= \lim_{i \rightarrow \infty} \frac{\frac{q_{i+2}^{(1)} - q_{i+1}^{(1)}}{\left(\frac{X_2}{X_1}\right)^i}}{\frac{q_{i+1}^{(1)} - q_i^{(1)}}{\left(\frac{X_2}{X_1}\right)^i}} \\
&= \frac{X_2 - \frac{X_2^2}{X_1}}{X_1 - X_2} \\
&= \frac{(X_1 - X_2) \left(\frac{X_2}{X_1}\right)}{(X_1 - X_2)} \\
&= \frac{X_2}{X_1} .
\end{aligned}$$

Eliminating X_1 we have

$$\begin{aligned}
\left(\lim_{i \rightarrow \infty} \frac{e_{i+1}^{(2)}}{e_i^{(2)}} \right) X_1 &= X_2 \\
\left(\lim_{i \rightarrow \infty} \frac{e_{i+1}^{(2)}}{e_i^{(2)}} \right) \left(\lim_{i \rightarrow \infty} q_i^{(1)} \right) &= X_2 .
\end{aligned}$$

But

$$\lim_{i \rightarrow \infty} q_i^{(1)} = \lim_{i \rightarrow \infty} q_{i+1}^{(1)} .$$

Hence

$$\lim_{i \rightarrow \infty} \frac{e_{i+1}^{(2)}}{e_i^{(2)}} q_{i+1}^{(1)} = X_2.$$

Further, let

$$q_{i+1}^{(2)} = \frac{e_{i+1}^{(2)}}{e_i^{(2)}} q_{i+1}^{(1)}.$$

Then we have the following:

$$\lim_{i \rightarrow \infty} q_i^{(1)} = X_1$$

$$\lim_{i \rightarrow \infty} q_{i+1}^{(2)} = \lim_{i \rightarrow \infty} q_i^{(2)} = X_2.$$

Let

$$e_i^{(3)} = q_{i+1}^{(2)} - q_i^{(2)} + e_i^{(2)}.$$

Then as before

$$\lim_{i \rightarrow \infty} \frac{e_{i+1}^{(3)}}{e_i^{(3)}} = \frac{X_3}{X_2}$$

$$\lim_{i \rightarrow \infty} \frac{e_{i+1}^{(3)}}{e_i^{(3)}} q_{i+1}^{(2)} = X_3$$

$$\frac{e_{i+1}^{(3)}}{e_i^{(3)}} q_{i+1}^{(2)} = q_{i+1}^{(3)}.$$

Thus

$$\lim_{i \rightarrow \infty} q_{i+1}^{(3)} = \lim_{i \rightarrow \infty} q_i^{(3)} = X_3.$$

Continuing by induction, the following results are established:

1. The rhombus rules have been obtained. They are

$$q_{n+1}^{(K)} = (e_n^{(K+1)} - e_n^{(K)}) + q_n^{(K)} \quad (K = 1, 2, \dots, N)$$

$$e_{n+1}^{(K)} = \frac{q_{n+1}^{(K)}}{q_{n+1}^{(K-1)}} e_n^{(K)} \quad (K = 2, 3, \dots, N+1)$$

where $n = 1, 2, 3, \dots$.

2. $\lim_{n \rightarrow \infty} q_n^{(K)} = X_K.$

2. Existence of the Q-D Scheme

The Q-D scheme described previously may fail to exist if one or more of the divisors $q_{n+1}^{(K-1)}$ of the rhombus rules above is zero. It appears to be difficult to give explicit necessary and sufficient conditions for existence of the scheme in terms of the polynomial $P(X)$. One sufficient condition in terms of the zeros of $P(X)$, given in [3, P. 165], is that the zeros X_1, X_2, \dots, X_N are simple and have distinct absolute values such that

$$|X_1| > |X_2| > \dots > |X_N| > 0.$$

3. Convergence Properties

Let the zeros X_1, X_2, \dots, X_N of $P(X)$ be numbered in decreasing

magnitude; that is,

$$|x_1| \geq |x_2| \geq \dots \geq |x_N|.$$

Then [10, P. 571] gives the following theorems.

Theorem 6.1. For every K such that $|x_{K-1}| > |x_K| > |x_{K+1}|$,

$$\lim_{n \rightarrow \infty} q_n^{(K)} = x_K ;$$

that is, the K^{th} column of the array converges to the K^{th} zero of $P(X)$.

The proof of this theorem has already been established in the derivation of the algorithm.

Theorem 6.2. For every K such that $|x_K| > |x_{K+1}|$

$$\lim_{n \rightarrow \infty} e_n^{(K)} = 0.$$

Proof: From the rhombus rules for $K = 1$ we obtain

$$q_{n+1}^{(1)} = e_n^{(2)} - e_n^{(1)} + q_n^{(1)}$$

$$\lim_{n \rightarrow \infty} e_n^{(2)} = \lim_{n \rightarrow \infty} q_{n+1}^{(1)} - \lim_{n \rightarrow \infty} q_n^{(1)} + \lim_{n \rightarrow \infty} e_n^{(1)}.$$

But $e_n^{(1)} = 0$ for all n and $\lim_{n \rightarrow \infty} q_n^{(K)} = x_K$ implies that

$$\lim_{n \rightarrow \infty} e_n^{(2)} = x_1 - x_1 + 0 = 0.$$

By induction we conclude that $\lim_{n \rightarrow \infty} e_n^{(K)} = 0$.

To this point, the existence of the scheme and convergence properties have been established for the case where the coefficients are non-zero and the zeros are simple and satisfy

$$|X_1| > |X_2| > \dots > |X_N| > 0.$$

The following observations should be noted:

1. The e-columns which approach zero (0) as a limit divide the Q-D scheme into subtables.
2. All zeros whose subscripts agree with the superscripts of the q-columns in a subtable have the same modulus.

Thus, if zero X_K is the only zero of its modulus, then

$$\lim_{n \rightarrow \infty} e_n^{(K)} = 0, \quad \lim_{n \rightarrow \infty} e_n^{(K+1)} = 0, \quad \lim_{n \rightarrow \infty} q_n^{(K)} = X_K.$$

4. Zeros of Equal Magnitude

Consider the case where M zeros, $X_{K+1}, X_{K+2}, \dots, X_{K+M}$, have the same modulus; that is,

$$|X_K| > |X_{K+1}| = |X_{K+2}| = \dots = |X_{K+M}| > |X_{K+M+1}|.$$

As noted above, the q-columns $q_n^{(K+1)}, q_n^{(K+2)}, \dots, q_n^{(K+M)}$ will be in the same subtable and the e-columns within that subtable, $e_n^{(K+2)}, \dots, e_n^{(K+M)}$, will not tend to zero. The zeros $X_{K+1}, X_{K+2}, \dots, X_{K+M}$ are then obtained from the following theorem given in [3, P. 167]. A proof of this result can be found in [12, PP. 42-43].

Theorem 6.3. Suppose $P(X)$ has M zeros of equal moduli satisfying

$$|x_K| > |x_{K+1}| = |x_{K+2}| = \dots = |x_{K+M}| > |x_{K+M+1}|.$$

Construct sequences of polynomials $P_n^{(j)}(X)$, $j = K, K+1, \dots, K+M$ by means of the following recurrence relations:

$$P_n^{(K)}(X) = 1, \quad n = 1, 2, 3, \dots$$

$$P_n^{(i)}(X) = X P_{n+1}^{(i-1)}(X) - q_n^{(i)} P_n^{(i-1)}(X) \quad i = K+1, \dots, K+M$$

$$n = 1, 2, \dots$$

Then limit $\lim_{n \rightarrow \infty} P_n^{(K+M)}(X) = (X - x_{K+1})(X - x_{K+2}) \dots (X - x_{K+M})$; that is, the coefficients of the polynomials $P_n^{(K+M)}(X)$ tend, as $n \rightarrow \infty$, to the coefficients of the polynomial with zeros $x_{K+1}, x_{K+2}, \dots, x_{K+M}$ and leading coefficient 1.

These polynomials can be thought of as being arranged in a two dimensional scheme, similar to the Q-D scheme as follows, where the $q_n^{(j)}$'s are obtained from the Q-D scheme.

The practical application of this theorem is as follows. Construct the Q-D scheme for n sufficiently large to determine convergence or non-convergence of the q -columns; that is, to divide the Q-D scheme into subtables. Convergence of a q -column corresponds to a zero of isolated absolute value. Divergence of M adjacent q -columns indicates that there are M zeros of equal moduli. Also the e -columns which converge to zero divide the scheme into subtables. Beginning with the first divergent q -column, one constructs the sequences of polynomials

$$P_n^{(K)}(X), P_n^{(K+1)}(X), \dots, P_n^{(K+M)}(X).$$

The roots X_{K+1}, \dots, X_{K+M} are then found as the zeros of

$$\lim_{n \rightarrow \infty} P_n^{(K+M)}(X).$$

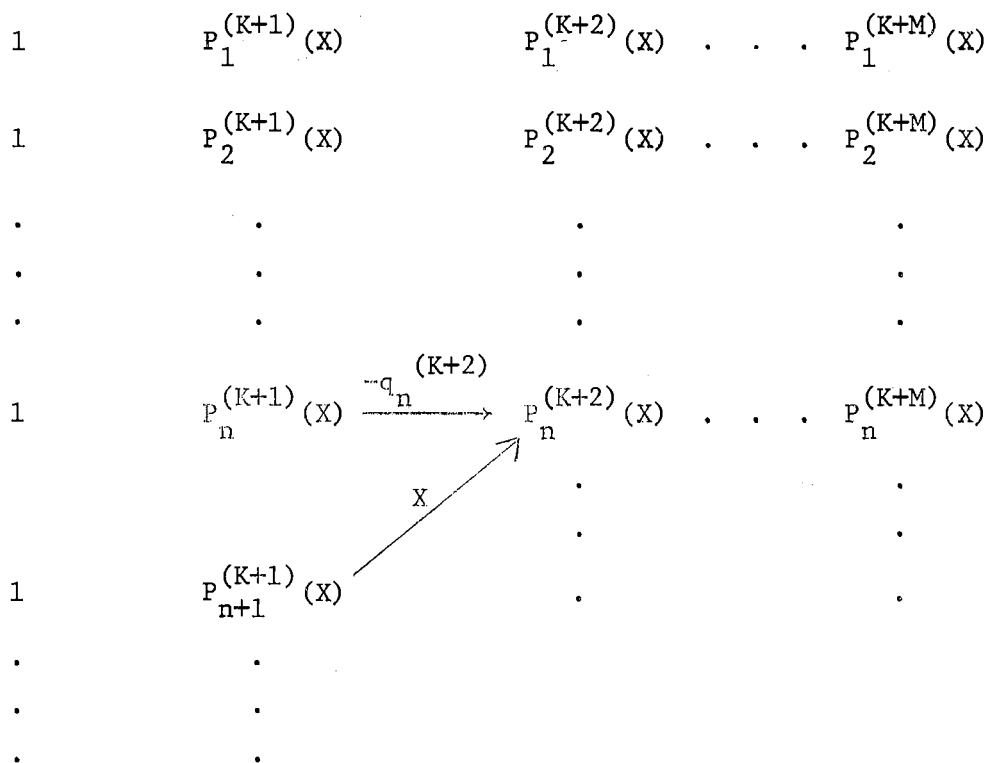


Figure 3.3. Scheme for the Polynomials of Theorem 6.3

5. Zero Coefficients of the Polynomial $P(X)$

Finally consider the case in which some of the coefficients a_2, a_3, \dots, a_N of $P(X)$ are zero. Let $X^* = X - a$, $a \neq 0$, for some suitably chosen a (usually $a = 1$). Construct the polynomial of degree

N

$$\begin{aligned}
 P^*(X^*) &= P(a + X^*) \\
 &= P(a) + \frac{P'(a)}{1!} X^* + \frac{P''(a)}{2!} X^{*2} + \dots + \frac{P^{(N)}(a)}{N!} X^{*N}
 \end{aligned}$$

whose coefficients are calculated by the following theorem, given in [3, PP. 54-56], which repeatedly uses Horner's method to obtain the coefficients of the Taylor series expansion of $P(X)$ about the point $X = a$.

Theorem 6.4. Given the finite sequence of numbers $b_1^{(1)}, b_2^{(1)}, \dots, b_{N+1}^{(1)}$, and the number a , generate the sequences $\{b_n^{(K)}\}$ for $K = 2, 3, \dots, N+2$ recursively by:

$$b_1^{(K)} = b_1^{(1)}$$

$$b_j^{(K)} = b_j^{(K-1)} + ab_{j-1}^{(K)}, \quad j = 2, 3, \dots, N+3-K.$$

Let $P_n(X) = b_1^{(1)} X^n + b_2^{(1)} X^{n-1} + \dots + b_n^{(1)} X + b_{n+1}^{(1)}$ ($n = 0, 1, \dots, N$).

Then the numbers $b_n^{(K)}$ determined above satisfy

$$\begin{aligned}
 b_j^{(K)} &= \frac{1}{(K-2)!} P_{j+K-3}^{(K-2)}(a), \quad j = 2, \dots, N+3-K \\
 &K = 2, \dots, N+2
 \end{aligned}$$

where $P^{(0)}(a) = P(a)$.

The above theorem indicates the following array for $N = 4$.

$b_1^{(1)}$	$b_2^{(1)}$	$b_3^{(1)}$	$b_4^{(1)}$	$b_5^{(1)}$
$b_1^{(2)}$	$b_2^{(2)}$	$b_3^{(2)}$	$b_4^{(2)}$	<u>$b_5^{(2)}$</u>
$b_1^{(3)}$	$b_2^{(3)}$	$b_3^{(3)}$	<u>$b_4^{(3)}$</u>	
$b_1^{(4)}$	$b_2^{(4)}$	<u>$b_3^{(4)}$</u>		
$b_1^{(5)}$	<u>$b_2^{(5)}$</u>			
$b_1^{(6)}$				

Figure 3.4. Scheme for Removing Zero Coefficients

The coefficients of the new polynomial $P^*(X^*)$ are the diagonal elements underlined in the scheme above; that is,

$$P^*(X^*) = b_1^{(6)} X^{*4} + b_2^{(5)} X^{*3} + b_3^{(4)} X^{*2} + b_4^{(3)} X^* + b_5^{(2)}$$

where

$$P(X) = b_1^{(1)} X^4 + b_2^{(1)} X^3 + b_3^{(1)} X^2 + b_4^{(1)} X + b_5^{(1)}.$$

The coefficients of $P^*(X^*)$ will all be non-zero for sufficiently small values of a , since if a is not a zero of $P(X)$ then the first N derivatives of $P(X)$ exist and are not zero at $X = a$. Once the zeros, $\{X_K^*\}_{K=1}^4$, of $P^*(X^*)$ have been computed, the zeros, $\{X_K\}_{K=1}^4$, of $P(X)$ are found from the formula $X_K = X_K^* + a$. As an example of the above theorem, let

$$P(X) = 12X^4 - 3X^3 + 7X + 10.$$

The coefficient of the X^2 term is 0, so the Q-D scheme cannot be started. With $a = 1$ we have for the above scheme

12	-3	0	7	10
12	9	9	16	26
12	21	30	46	
12	33	63		
12	45			
12				

Then

$$P^*(X^*) = 12X^{*4} + 45X^{*3} + 63X^{*2} + 46X + 26.$$

Let the zeros of $P^*(X^*)$ be $X_1^*, X_2^*, X_3^*, X_4^*$. Then the zeros of $P(X)$ are $X_i = X_i^* + 1$, $i = 1, 2, 3, 4$. This concludes the example.

All necessary cases have thus been considered. They are:

1. distinct zeros
2. zeros of equal magnitude
3. zero coefficients of the polynomial $P(X)$.

Hence, the Quotient-Difference method can now be applied to the polynomial $P(X) = a_1X^N + a_2X^{N-1} + \dots + a_NX + a_{N+1}$ where $a_1 \neq 0 \neq a_{N+1}$.

6. Procedure for the Q-D Method

The sequence of basic steps performed by this method are as follows:

1. Given coefficients a_1, a_2, \dots, a_{N+1} of the N^{th} degree polynomial, $P(X)$, such that $a_1 \neq 0 \neq a_{N+1}$.
2. Determine if any of the coefficients in step 1 are equal to zero. If not, proceed to step 3. If one or more are zero, use Theorem 6.4 to construct a polynomial whose coefficients are all non-zero.
3. Begin the Q-D scheme by constructing the first two rows with the formulas:

$$q_1^{(1)} = -a_2/a_1$$

$$q_1^{(K)} = 0 \quad (K = 2, 3, \dots, N)$$

$$e_1^{(K)} = a_{K+1}/a_K \quad (K = 2, 3, \dots, N)$$

Set the first and last columns to zero with

$$e_n^{(1)} = e_n^{(N+1)} = 0.$$

4. Construct the Q-D scheme row after row by using the rhombus rules:

$$q_{n+1}^{(K)} = \left(e_n^{(K+1)} - e_n^{(K)} \right) + q_n^{(K)} \quad (K = 1, 2, \dots, N)$$

$$e_{n+1}^{(K)} = \frac{q_{n+1}^{(K)}}{q_{n+1}^{(K-1)}} e_n^{(K)} \quad (K = 2, 3, \dots, N).$$

5. Test for convergence of the e-columns. Either of the following tests may be used:

- a. $2|e_n^{(K)}| / (|q_n^{(K)} + q_n^{(K+1)}| + |q_{n-1}^{(K)} + q_{n-1}^{(K+1)}|)$

- b. $-\text{EPS} \left| \frac{e_n^{(K)}}{e_{25}^{(K)}} \right| -\text{EPS}$ where EPS is chosen sufficiently small so that the test expression will remain positive until the K^{th} e-column has sufficiently converged. n is the number of iterations and is an input value.
6. After the maximum number of iterations, determine which e-columns have not converged. Then divide the Q-D scheme into subtables. Proceeding from left to right in the scheme, do step 7 or step 8 according to the number of q-columns in the subtable until all subtables have been considered.
 7. If a subtable has two or more q-columns in it, use Theorem 6.3 to determine the zeros of that subtable.
 8. If a subtable contains only one q-column, then the final element in the q-column is the desired approximation of a zero of the polynomial.
 9. Check the root count. If not all roots have been found, divide out the zeros that have been found by successively deflating $P(X)$ using synthetic division. Shift the variable in the resulting polynomial for which no zeros were found by using Theorem 6.4. Then return to step 1. See Theorem 2.3 for the iteration formulas used to deflate a polynomial.
 10. If it was necessary to use Theorem 6.4 in either step 2 or 9, the zeros $\{X_K^*\}_{K=1}^N$ found are the zeros of the new polynomial $P^*(X^*)$. Use $X_K = X_K^* + a$ to find the zeros $\{X_K\}_{K=1}^N$ of $P(X)$.

7. Conclusion of the Method

As suggested earlier, it was decided to use the Q-D method for the purpose of producing good initial approximations to the zeros which would then be used with another method to produce final results. The following difficulty was encountered. In order to program the Q-D method to solve polynomials with maximum degree of twenty five, the size of core storage needed was considered and found to be quite large. For M roots of equal magnitude, Theorem 6.3 generates M sequences of polynomials having degrees $1, 2, 3, \dots, M$. The number of terms needed in each sequence to produce convergence is not known before hand. In addition, M terms of the first sequence are necessary to compute the first term of the M^{th} sequence since, as seen from the scheme, the sequences are generated in a triangular manner. Therefore, in order to get sufficient convergence of the M^{th} sequence of polynomials, the number of terms in the first few sequences could become quite large. For example, suppose in a polynomial of degree 25, 20 roots have equal moduli. Then Theorem 6.3 calls for 20 sequences of polynomials to be generated, the first of degree 1, the second of degree 2, ..., and the 20th of degree 20. Twenty terms of the first, 19 terms of the second, and so on, would be necessary to calculate the first term of the 20th sequence. Thus,

$$\sum_{I=2}^{21} I(22 - I)$$

locations would be needed to store all coefficients. In general, for M roots of equal moduli,

$$\sum_{I=2}^{M+1} I(M+2-I)$$

locations are needed to compute the first term of the M^{th} sequence. An attempt was made to decrease the number of locations needed for storing the polynomial sequences by computing the terms of each polynomial sequence, storing the last polynomial of each sequence and repeating in order to continue each sequence using the same storage area for each repetition. However, this process of transferring polynomials, storing polynomials, bringing in new q-columns from the Q-D scheme, and continuing the polynomial sequences until convergence was obtained, presented a programming problem of greater magnitude than had been anticipated. Thus, considering the immediate purpose of this method, to give good initial approximations to the zeros, the slow convergence, and the lack of time, it was felt unprofitable to complete the programming of the Q-D method. Also, storage location requirements for the elements of the e-columns and q-columns of the Q-D scheme could become quite large unless the scheme is generated a few rows at a time. To do this would further add to the problem of generating the polynomial sequences using the q-columns.

CHAPTER VII

CONCLUSION

In order to compare Newton's, Muller's, and the greatest common divisor (g.c.d.) methods, consider the polynomials as being divided into the following classes:

1. polynomials with all distinct roots
2. polynomials with multiple roots
3. polynomials with roots close together

By "close together" I mean distinct roots agreeing in at least the first three significant digits. These roots will not be included in class 1.

The comparisons in the following material, except where specifically noted, are results of tests made on the IBM S/360 mod. 50 computer which has a 32 bit word. The programs were successfully run on the CDC 6600 and the UNIVAC 1108 which have a 60 bit word and a 36 bit word respectively. It was noted that the UNIVAC 1108 is about 15 times faster than the IBM S/360 mod. 50. The CDC 6600 is faster than the UNIVAC 1108 but the difference is not as great as that between the UNIVAC 1108 and the IBM S/360 mod. 50.

1. Polynomials With all Distinct Roots

First consider the class of polynomials having distinct roots. Newton's method is particularly suited for this class of polynomials. Its quadratic convergence is very fast which can save time and money to

the user. The accuracy obtained is excellent as shown in exhibits A (12 sec.) and B (22 sec.) which present the zeros of a 15th degree polynomial in single precision and double precision, respectively. In most cases, the method produces convergence for almost any initial approximation given.

Muller's method also produces good results on this class of polynomials. The rate of convergence is, however, somewhat slower than Newton's method. This fact is especially significant when working with polynomials of high degree. The accuracy obtained by Muller's method is comparable to, but does not exceed that of Newton's method. In most cases, the accuracy of the two methods does not differ by more than one or two decimal places. Exhibits C (17 sec.) and D (28 sec.) show results of Muller's method for the polynomial of exhibits A and B. As in Newton's method, convergence is produced for almost any initial approximation given.

The g.c.d. method, whether used with Newton's or Muller's method as a supporting method on this class of polynomials, is no better than Newton's or Muller's method alone. The reason for this is that the greatest common divisor of the polynomial, $P(X)$, and its derivative is 1. Then $H(X) = P(X)/\text{g.c.d. } P(X) = P(X)$; that is, the polynomial solved by the supporting method is the same as the original polynomial. Thus, the g.c.d. method will require a considerable amount of extra time to perform useless calculations and will not produce better results than the supporting method used alone.

Thus, this class of polynomials presents no difficulty to any of these three methods. Newton's method, because of its speed, is therefore recommended.

2. Polynomials With Multiple Roots

Next consider the class of polynomials containing multiple roots. This class presents considerable difficulty for Newton's method, especially those polynomials containing roots of high multiplicity or containing a considerable number of multiple roots. The iteration formula for Newton's method is

$$X_{n+1} = X_n - P(X_n)/P'(X_n).$$

If c is a multiple root then $P(c) = P'(c) = 0$. Hence, as $X_n \rightarrow c$, $P(X_n) \rightarrow 0$ and $P'(X_n) \rightarrow 0$ and the iteration formula may be unstable, resulting in no convergence or bad accuracy. As the number of multiple roots increases, the polynomial becomes more ill-conditioned, convergence becomes more difficult, and accuracy is lost. Thus, the possibility of convergence decreases. This also holds true if the multiplicities of the roots are increased. The rate of convergence of Newton's method is much slower for multiple roots than for distinct roots. Exhibits E (38 sec.) and F (34 sec.) show a polynomial containing two multiple roots solved in single precision and double precision, respectively. Note the following from exhibit F:

1. Roots #2 and #3 are greatly improved by iterating on the original polynomial. Distinct roots are usually improved in this manner.
2. The time taken to solve this 6th degree equation with multiple roots is greater than the time taken by the same program to solve a 15th degree polynomial with all distinct roots (exhibit B).

3. Root #2 did not pass the convergence test after 200 iterations even though it was improved. This is probably due to the fact that the polynomial from which root 2 was extracted had only one multiple root but the original polynomial from which it was extracted the second time had two multiple roots; that is, the original polynomial is more ill-conditioned.
4. The accuracy of the roots before the attempt to improve accuracy is very poor. Root #2 is accurate to only three decimal places as compared to the 15 decimal places in exhibit B for distinct roots. Root #3 is especially bad, the imaginary part being accurate to only one decimal place.

As seen from these exhibits, single precision calculations do not produce good results when multiple roots are involved. The time for exhibit E was 38 seconds. Compare this with exhibit A for the 15th degree polynomial. Exhibit G (14 sec.) is a polynomial containing two multiple roots. Note the poor results obtained before the attempt to improve accuracy and the improvement afterward.

In most cases involving roots of high multiplicity or several multiple roots, Newton's method fails to determine the correct multiplicity of many of the roots even though all the roots have been obtained.

In many cases, Newton's method fails to converge altogether. The polynomial with roots (the number in parentheses indicates multiplicity) $2 + 2i$ (3), $1 + 2i$ (2), $-1 + .5i$ (3) could not be solved using Newton's

method with a maximum number of 200 iterations and a convergence requirement of 10^{-10} . This polynomial is given as number 25 in exhibit H using Muller's method. Observe that this polynomial is polynomial #53 of exhibit F with the multiplicity of root $-1 + .5i$ increased from 1 to 3.

Muller's method also encounters difficulty, although to a lesser degree than Newton's method, on this class of polynomials. In most cases, Muller's method produces convergence even when Newton's method completely fails. Newton's method completely failed for polynomial #25 but convergence was obtained using Muller's method as shown in exhibit H (40 sec.). The accuracy obtained by Muller's method is not good but usually better than Newton's method using the same convergence requirement. Compare exhibit J (15 sec.) with exhibit G. The rate of convergence of Muller's method is considerably slower for multiple roots than for distinct roots. However, for multiple roots, Muller's method is as fast or faster than Newton's. Newton's method appears to determine multiplicities more correctly than Muller's method. Note the following from exhibit J:

1. Roots #3, #4, and #5 are greatly improved by iterating on the original polynomial, especially the distinct root #4.
2. The accuracy obtained before the attempt to improve accuracy is very poor compared to the accuracy obtained with all distinct roots. The accuracy is, however, better than Newton's method (exhibit G).
3. The multiplicities of the roots are not determined correctly.

Polynomial #46 in single precision is given in exhibit I. Similarly as with Newton's method, an increase in the number of multiple roots or their multiplicities causes the polynomial to become more ill-conditioned. As a result, accuracy is lost.

The g.c.d. method is perfectly suited for polynomials with multiple roots. All multiple roots are removed leaving only a polynomial of class 1 (all distinct roots) to be solved. This indicates that best results should be obtained by using Newton's method as the supporting method, since Newton's method enjoys the advantage of speed over Muller's method for distinct roots. This has indeed proved to be true. The accuracy of the roots obtained decreases, somewhat, when the number of multiple roots is increased. This is due to accuracy lost in computing the g.c.d. and the quotient polynomial and not as a result of the supporting method. The accuracy obtained using each supporting method is about the same.

Multiplicities are determined with excellent accuracy. The g.c.d. method is not as sensitive to roots of high multiplicity or polynomials containing a large number of multiple roots as are both Newton's and Muller's. The g.c.d. method is faster than either Newton's or Muller's because multiple roots greatly slow the rate of convergence of the latter two. Exhibits K (2 sec.) and L (3 sec.) show polynomial #25 for which Newton's method gave no convergence and Muller's method gave poor convergence. Note that the execution time for double precision is considerably less than for Muller's method alone (exhibit H). The multiplicities are correct. Exhibit M (3 sec.) is a polynomial containing two roots each of multiplicity 6. Newton's method used alone did not produce convergence to any root of this polynomial and Muller's method

gave bad accuracy. Note the accuracy and speed of the g.c.d. method on this polynomial. Exhibit N (12 sec.) contains 6 multiple roots. Observe that the accuracy is not quite as good as in the preceding two exhibits because there are more multiple roots. Note also that there is not much improvement in the roots after the attempt to improve accuracy. This is characteristic of Newton's and Muller's accuracy on distinct roots, and further supports the conclusion that the loss of accuracy is a result of computing the g.c.d. polynomial. Again the multiplicities are all correct.

The accuracy of multiple roots obtained by the CDC 6600 is considerably better than the IBM S/360 mod. 50 while that obtained by the UNIVAC 1108 is only a little better than the IBM S/360 mod. 50.

Therefore, for polynomials with multiple roots, the order in which the three methods are recommended, beginning with the best is: g.c.d. with Newton's, Muller's, Newton's.

Since multiple roots are obtained with less accuracy than distinct roots, lowering the convergence requirement produced convergence in many cases where the higher convergence requirement fails to produce convergence. The accuracy is usually fair for a polynomial with few multiple roots and decreases as the number of multiple roots increases. In most cases, the accuracy is sufficient to give a good approximation of the roots. Exhibit O shows polynomial #25 as a result of using the convergence requirement of 10^{-5} . The roots after the attempt to improve accuracy are accurate to at least 4 decimal places with some gaining accuracy to 6 or 7 decimal places. Recall that the convergence requirement of 10^{-10} produced no convergence.

3. Polynomials With Roots Close Together

Polynomials with distinct roots close together present considerable difficulty for all three methods. Ability to converge and the accuracy obtained depends on the number of these roots involved and their closeness. If care is not taken in choosing the multiplicity requirement, roots very close together may not be separated and, as a result, will be obtained as multiple roots. Exhibit P (5 sec.) shows what happens when the multiplicity requirement is too small. Note the following from this exhibit:

1. Root $.103 + .103i$ was not obtained.
2. Root #2 has multiplicity 2 which is incorrect.
3. Root #3, before the attempt to improve accuracy, is entirely incorrect. This is the result of deflating by an incorrect root. The correct root was obtained by iterating on the original polynomial. This emphasizes the necessity of observing the results both before and after the attempt to improve accuracy.

When the results indicate that roots close together are present, the multiplicity requirement should be sufficiently increased to separate the roots. This was done on the above polynomial and the results and are indicated in exhibit Q (5 sec.). Note the reasonable accuracy of these roots all of which agree in the first two significant digits but differ in the third. Newton's method failed to converge to any root on polynomial #33 with roots, $1.010 + 1.020i$, $1.011 + 1.021i$, $1.012 + 1.022i$, $1.013 + 1.023i$ within 200 iterations and using a

convergence requirement of 10^{-10} . Note that these roots agree in the first three significant digits but differ in the fourth.

Polynomial #33 was solved with excellent accuracy by the CDC 6600. The test for convergence was 10^{-14} and the test for multiplicities was 10^{-8} . The roots, as found, were

$$\text{Root (1)} = 1.0100000000000000D+00 + 1.0200000000000000D+00 \text{ I}$$

$$\text{Root (2)} = 1.0110000000000000D+00 + 1.0210000000000000D+00 \text{ I}$$

$$\text{Root (3)} = 1.0130000000000000D+00 + 1.0230000000000000D+00 \text{ I}$$

$$\text{Root (4)} = 1.0120000000000000D+00 \quad 1.0220000000000000D+00 \text{ I}$$

This polynomial was also solved by the UNIVAC 1108 using a test for convergence of 10^{-8} and a test for multiplicities of 10^{-6} . The results were

$$\text{Root (1)} = .1010000000884303+001 + .1019999999482552+001 \text{ I}$$

$$\text{Root (2)} = .1011000000380731+001 + .10209999998517433+001 \text{ I}$$

$$\text{Root (3)} = .10129999998246209+001 + .10229999998494446+001 \text{ I}$$

$$\text{Root (4)} = .1012000001356113+001 + .1021999999747005+001 \text{ I}$$

Compare these results with exhibit V.

Muller's method is recommended for this class of polynomials because it produces convergence in most cases where Newton's method fails. When both methods produce convergence, the accuracy obtained by Muller's method is as good or better than that obtained by Newton's. Compare exhibits R (20 sec.) and Q. Again, a multiplicity requirement chosen too small produces incorrect results as in Newton's method. Exhibit S gives polynomial #33 solved by Muller's method. Observe that roots #2 and #4 obtained before the attempt to improve accuracy

converged to the same root during the attempt to improve accuracy. Recall that Newton's method failed to converge to any root of this polynomial.

The g.c.d. method, as in the case of all distinct roots, is no better than the supporting method used. In many cases where very close roots are involved, the incorrect greatest common divisor may be obtained by treating these as multiple roots. Thus, incorrect results will be obtained. This occurs when the requirement check for zero given to the GCD routine is too low. This requirement can be increased to correct the error. However, if this requirement is made too high when multiple roots are involved, an incorrect g.c.d. may be obtained. Also, a multiplicity requirement too low will produce incorrect results. Exhibit T shows the result of both the epsilon check for zero in subroutine GCD and the epsilon check for multiplicities being too low. Both the root and the multiplicity are incorrect. These requirements were increased and the result given as exhibit U (7 sec.). Correct results were obtained. Thus, as a result of having to carefully choose both the epsilon check in subroutine GCD and the multiplicity requirement, this method is not recommended for this class of polynomials.

As in the case of multiple roots, the convergence requirement may be lowered to obtain convergence in most cases where a high convergence requirement causes failure to converge. Convergence was obtained for polynomial #33 (exhibit V (4 sec.)) by lowering the convergence requirement from 10^{-10} to 10^{-8} .

Another technique for obtaining convergence on ill-conditioned polynomials is to add some distinct roots which are not close together. This is accomplished by multiplying the polynomial by several linear

factors. As an example of this technique, consider polynomial #33 on which Newton's method failed to converge. After multiplying the polynomial by the factors $(X - 4 + 2i)$ and $(X - 3 - 5i)$, convergence was obtained as indicated in exhibit W (24 sec.). The number of distinct roots added to obtain convergence depends on how bad the polynomial is ill-conditioned.

For polynomials containing a combination of these three types of roots, the method used to obtain best accuracy depends on the number of each type involved. The g.c.d. method is usually best because distinct and multiple roots are obtained with good accuracy. Those roots obtained with poor accuracy are probably roots very close together. Increasing the test for zero in subroutine GCD may separate these close roots but multiple roots may not be obtained with good accuracy. If roots close together are present, another method may be used to separate these roots. Exhibit X (6 sec.) gives a polynomial with two multiple roots, two distinct roots not close together, and two distinct roots close together. This polynomial was first solved using the g.c.d. method with Newton's method as the supporting method. Note that all roots except the roots close together are reasonably accurate and their multiplicities correct. Using the results after the attempt to improve accuracy as initial approximations, this polynomial was solved by Muller's method as indicated in exhibit Y (39 sec.). The multiplicity requirement was increased considerably above normal to prevent any root, especially roots very close together, from being obtained as multiple roots. This tends to separate the roots close together. Note the following:

1. The two roots close together were obtained very accurately before the attempt to improve accuracy but both converged to the same root later.
2. The accuracy of all roots is good.

Exhibit Z (18 sec.) presents the results of solving this polynomial by Muller's method but letting the program generate its own initial approximations. Note that the accuracy is not as good as that obtained using the initial approximations close to the roots.

Thus, each method is particularly suited for a different type of polynomial. Newton's is best for polynomials with distinct roots not too close together. The g.c.d. method is superior for polynomials having multiple roots, while Muller's method is recommended for extracting roots very close together.

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 7 OF DEGREE 15

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.3000000E 01 + 0.0000000E 00 I
P(2) = -0.1789999E 02 + 0.0000000E 00 I
P(3) = 0.2010001E 02 + -0.6575999E 02 I
P(4) = 0.1745000E 03 + 0.2843679E 03 I
P(5) = -0.7594199E 03 + 0.3118081E 03 I
P(6) = 0.8274360E 03 + -0.3069040E 04 I
P(7) = 0.1329084E 04 + 0.4710625E 04 I
P(8) = -0.5611859E 04 + -0.1674576E 04 I
P(9) = 0.7224758E 04 + -0.1548288E 04 I
P(10) = -0.2276992E 04 + 0.3046320E 04 I
P(11) = -0.1241472E 04 + -0.4097281E 04 I
P(12) = 0.5402801E 04 + 0.1263488E 04 I
P(13) = -0.6468336E 04 + 0.1236480E 04 I
P(14) = 0.1467456E 04 + 0.2272000E 02 I
P(15) = -0.1077120E 03 + -0.5475840E 03 I
P(16) = 0.3456000E 02 + 0.1267200E 03 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.10E-04
TEST FOR MULTIPLICITIES. 0.10E-01
RADIUS TO START SEARCH. 0.00E 00
RADIUS TO END SEARCH. 0.00E 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ZEROS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.3000001E 00 + 0.6291682E-08 I	1	0.4829629E 00 + 0.1294095E 00 I
ROOT(2) = 0.2000000E 00 + -0.2000002E 00 I	1	0.7071067E 00 + 0.7071068E 00 I
ROOT(3) = -0.2000002E 00 + 0.2000002E 00 I	1	0.3882295E 00 + 0.1448888E 01 I
ROOT(4) = 0.1789849E-06 + 0.1000000E 01 I	1	-0.5176364E 00 + 0.1931851E 01 I
ROOT(5) = -0.1000000E 01 + -0.3287970E-06 I	1	-0.1767765E 01 + 0.1767768E 01 I
ROOT(6) = -0.1000000E 01 + -0.1000002E 01 I	1	-0.2897776E 01 + 0.7764602E 00 I
ROOT(7) = -0.2310580E-06 + -0.1000000E 01 I	1	-0.3380741E 01 + -0.9058627E 00 I
ROOT(8) = -0.1999997E 01 + -0.3000000E 01 I	1	-0.2828430E 01 + -0.2828424E 01 I
ROOT(9) = 0.9999008E 00 + -0.2495819E-03 I	1	-0.1164691E 01 + -0.4346664E 01 I
ROOT(10) = 0.2000046E 01 + -0.1000127E 01 I	1	0.1294087E 01 + -0.4829631E 01 I
ROOT(11) = 0.2000143E 01 + 0.3578651E-03 I	1	0.3889081E 01 + -0.3889092E 01 I
ROOT(12) = 0.2999989E 01 + -0.5277649E-04 I	1	0.5795551E 01 + -0.1552923E 01 I
ROOT(13) = 0.3999991E 01 + 0.4000009E 01 I	1	0.6278520E 01 + 0.1682312E 01 I
ROOT(14) = 0.9999154E 00 + 0.1000066E 01 I	1	SOLVED BY DIRECT METHOD
ROOT(15) = -0.3333335E 01 + -0.1112619E-05 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

Exhibit A.

ZEROS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.3000001E 00 + -0.9981129E-08 I	1	0.4829629E 00 + 0.1294095E 00 I
ROOT(2) = 0.2000000E 00 + -0.2000000E 00 I	1	0.7071067E 00 + 0.7071068E 00 I
ROOT(3) = -0.2000000E 00 + 0.2000000E 00 I	1	0.3882295E 00 + 0.1448888E 01 I
ROOT(4) = 0.1446784E-07 + 0.1000000E 01 I	1	-0.5176364E 00 + 0.1931851E 01 I
ROOT(5) = -0.1000000E 01 + -0.2083993E-08 I	1	-0.1767765E 01 + 0.1767768E 01 I
ROOT(6) = -0.1000000E 01 + -0.1000000E 01 I	1	-0.2897776E 01 + 0.7764602E 00 I
ROOT(7) = 0.1241104E-06 + -0.1000000E 01 I	1	-0.3380741E 01 + -0.9058627E 00 I
ROOT(8) = -0.2000000E 01 + -0.3000000E 01 I	1	-0.2828430E 01 + -0.2828424E 01 I
ROOT(9) = 0.1000000E 01 + 0.1584821E-06 I	1	-0.1164691E 01 + -0.4346664E 01 I
ROOT(10) = 0.1999998E 01 + -0.1000001E 01 I	1	0.1294087E 01 + -0.4829631E 01 I
ROOT(11) = 0.2000008E 01 + 0.8055767E-06 I	1	0.3889081E 01 + -0.3889092E 01 I
ROOT(12) = 0.2999995E 01 + 0.2530969E-05 I	1	0.5795551E 01 + -0.1552923E 01 I
ROOT(13) = 0.4000000E 01 + 0.4000002E 01 I	1	0.6278520E 01 + 0.1682312E 01 I
ROOT(14) = 0.9999998E 00 + 0.1000000E 01 I	1	SOLVED BY DIRECT METHOD
ROOT(15) = -0.3333334E 01 + 0.4381855E-06 I	1	SOLVED BY DIRECT METHOD

Exhibit A. Roots Are: $-1 - i$, $1 + i$, $-2 - 3i$, $2 - i$, 3 , 2 , i , $-i$, $-10/3$, $.3$, -1 , 1 , $4 + 4i$, $-2 + .2i$, $.2 - .2i$.

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 7 OF DEGREE 15

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.3000000000000000 01 + 0.0000000000000000 00 I
P(2) = -0.1790000000000000 02 + -0.0000000000000000 00 I
P(3) = 0.2010000000000000 02 + -0.6576000000000000 02 I
P(4) = 0.1745000000000000 03 + 0.2843680000000000 03 I
P(5) = -0.7594200000000000 03 + 0.3118080000000000 03 I
P(6) = 0.8274360000000000 03 + -0.3069040000000000 04 I
P(7) = 0.1329084000000000 04 + 0.4710624000000000 04 I
P(8) = -0.5611860000000000 04 + -0.1674576000000000 04 I
P(9) = 0.7224756000000000 04 + -0.1548288000000000 04 I
P(10) = -0.2276992000000000 04 + 0.3046320000000000 04 I
P(11) = -0.1241472000000000 04 + -0.4097280000000000 04 I
P(12) = 0.5402800000000000 04 + 0.1263488000000000 04 I
P(13) = -0.6468336000000000 04 + 0.1236480000000000 04 I
P(14) = 0.1467456000000000 04 + 0.2272000000000000 02 I
P(15) = -0.1077120000000000 03 + -0.5475840000000000 03 I
P(16) = 0.3456000000000000 02 + 0.1267200000000000 03 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
R00T(1) = 0.3000000000000000 00 + 0.1609358115166531D-16 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
R00T(2) = 0.2000000000000000 00 + -0.2000000000000000 00 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
R00T(3) = -0.2000000000000000 00 + 0.2000000000000000 00 I	1	0.3882284792654056D 00 + 0.1448888763117193D 01 I
R00T(4) = 0.2485026109655803D-17 + 0.1000000000000000 01 I	1	-0.5176382551966724D 00 + 0.1931851608368755D 01 I
R00T(5) = -0.1000000000000000 01 + -0.5321953706459946D-16 I	1	-0.1767767147080701D 01 + 0.1767766758852015D 01 I
R00T(6) = -0.1000000000000000 01 + -0.9999999999999998D 00 I	1	-0.2897777583074990D 01 + 0.7764567463987070D 00 I
R00T(7) = -0.3106918759280349D-15 + -0.9999999999999994D 00 I	1	-0.3380740248331229D 01 + -0.9058671940816160D 00 I
R00T(8) = -0.2000000000000000 01 + -0.3000000000000000 01 I	1	-0.2828426607107896D 01 + -0.2828427642384390D 01 I
R00T(9) = 0.1000000000000016D 01 + -0.1009764996352100D-12 I	1	-0.1164684801399899D 01 + -0.4346666459873368D 01 I
R00T(10) = 0.2000000000000040D 01 + -0.1000000000000032D 01 I	1	0.1294096345098645D 01 + -0.4829628831453027D 01 I
R00T(11) = 0.1999999999999980D 01 + 0.138550784722202D-12 I	1	0.3889088292979509D 01 + -0.3889086300072258D 01 I
R00T(12) = 0.3000000000000001D 01 + -0.1417549157449219D-13 I	1	0.5795555393512303D 01 + -0.1552912644268974D 01 I
R00T(13) = 0.399999999999998D 01 + 0.4000000000000003D 01 I	1	0.6278517357734125D 01 + 0.1682325708247752D 01 I
R00T(14) = 0.9999999999999606D 00 + 0.1000000000000005D 01 I	1	SOLVED BY DIRECT METHOD
R00T(15) = -0.333333333333334D 01 + -0.1850371707708594D-15 I	1	SOLVED BY DIRECT METHOD

Exhibit B.

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.3000000000000000 00 + 0.3328943537705913D-16 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = 0.2000000000000000 00 + -0.2000000000000000 00 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
ROOT(3) = -0.2000000000000000 00 + 0.2000000000000000 00 I	1	0.3882284792654056D 00 + 0.1448888763117193D 01 I
ROOT(4) = -0.1126043043762753D-17 + 0.1000000000000000 01 I	1	-0.5176382551966724D 00 + 0.1931851608368755D 01 I
ROOT(5) = -0.1000000000000000 01 + -0.1882478030550796D-16 I	1	-0.1767767147080701D 01 + 0.1767766758852015D 01 I
ROOT(6) = -0.1000000000000000 01 + -0.1000000000000000 01 I	1	-0.2897777583074990D 01 + 0.7764567463987070D 00 I
ROOT(7) = 0.6131869319379140D-16 + -0.1000000000000000 01 I	1	-0.3380740248331229D 01 + -0.9058671940816160D 00 I
ROOT(8) = -0.2000000000000000 01 + -0.3000000000000000 01 I	1	-0.2828426607107896D 01 + -0.2828427642384390D 01 I
ROOT(9) = 0.1000000000000000 01 + 0.7287765480895064D-16 I	1	-0.1164684801399899D 01 + -0.4346666459873368D 01 I
ROOT(10) = 0.2000000000000000 01 + -0.9999999999999999D 00 I	1	0.1294096345098645D 01 + -0.4829628831453027D 01 I
ROOT(11) = 0.2000000000000000 01 + -0.3968039620811357D-14 I	1	0.3889088292979509D 01 + -0.3889086300072258D 01 I
ROOT(12) = 0.2999999999999997D 01 + 0.2896889630437848D-14 I	1	0.5795555393512303D 01 + -0.1552912644268974D 01 I
ROOT(13) = 0.4000000000000000 01 + 0.4000000000000000 01 I	1	0.6278517357734125D 01 + 0.1682325708247752D 01 I
ROOT(14) = 0.9999999999999950 00 + 0.1000000000000000D 01 I	1	SOLVED BY DIRECT METHOD
ROOT(15) = -0.3333333333333334D 01 + -0.8998063670517638D-16 I	1	SOLVED BY DIRECT METHOD

Exhibit B. Roots Are: $-1 - i$, $1 + i$, $-2 - 3i$, $2 - i$, 3 , 2 , i , $-i$, $-10/3$, $.3$, -1 , 1 , $4 + 4i$, $-.2 + .2i$, $.2 - .2i$.

MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 7 OF DEGREE 15

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.3000000E 01 + 0.0000000E 00 I
P(2) = -0.1789999E 02 + 0.0000000E 00 I
P(3) = 0.2010001E 02 + -0.6575999E 02 I
P(4) = 0.1745000E 03 + 0.2843679E 03 I
P(5) = -0.7594199E 03 + 0.3118081E 03 I
P(6) = 0.8274360E 03 + -0.3069040E 04 I
P(7) = 0.1329084E 04 + 0.4710625E 04 I
P(8) = -0.5611859E 04 + -0.1674576E 04 I
P(9) = 0.7224758E 04 + -0.1548288E 04 I
P(10) = -0.2276992E 04 + 0.3046320E 04 I
P(11) = -0.1241472E 04 + -0.4097281E 04 I
P(12) = 0.5402801E 04 + 0.1263488E 04 I
P(13) = -0.6468336E 04 + 0.1236480E 04 I
P(14) = 0.1467456E 04 + 0.2272000E 02 I
P(15) = -0.1077120E 03 + -0.5475840E 03 I
P(16) = 0.3456000E 02 + 0.1267200E 03 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.10E-04
TEST FOR MULTIPLICITIES. 0.10E-01
RADIUS TO START SEARCH. 0.00E 00
RADIUS TO END SEARCH. 0.00E 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.3000001E 00 + 0.2621618E-08 I	1	0.4829629E 00 + 0.1294095E 00 I
ROOT(2) = -0.2000000E 00 + 0.2000000E 00 I	1	0.7071067E 00 + 0.7071068E 00 I
ROOT(3) = 0.9999995E 00 + 0.1000000E 01 I	1	0.3882295E 00 + 0.1448888E 01 I
ROOT(4) = 0.9999982E 00 + -0.3130342E-05 I	1	-0.5176364E 00 + 0.1931851E 01 I
ROOT(5) = -0.1000000E 01 + -0.1000000E 01 I	1	-0.1767765E 01 + 0.1767768E 01 I
ROOT(6) = -0.3333335E 01 + -0.2337438E-06 I	1	-0.2897776E 01 + 0.7764602E 00 I
ROOT(7) = -0.1000060E 01 + -0.2979267E-04 I	1	-0.3880741E 01 + -0.9058627E 00 I
ROOT(8) = -0.2000000E 01 + -0.3000000E 01 I	1	-0.2828430E 01 + -0.2828424E 01 I
ROOT(9) = 0.2000399E 01 + -0.2037037E-03 I	1	-0.1164691E 01 + -0.4346664E 01 I
ROOT(10) = 0.2999865E 01 + -0.7479227E-04 I	1	0.1294087E 01 + -0.4829631E 01 I
ROOT(11) = 0.1622662E-04 + 0.9999509E 00 I	1	0.3889081E 01 + -0.3889092E 01 I
ROOT(12) = -0.2123728E-03 + -0.1000231E 01 I	1	0.5795551E 01 + -0.1552923E 01 I
ROOT(13) = 0.3999983E 01 + 0.4000027E 01 I	1	0.6278520E 01 + 0.1682312E 01 I
ROOT(14) = 0.1999767E 01 + -0.1000011E 01 I	1	SOLVED BY DIRECT METHOD
ROOT(15) = 0.2002156E 00 + -0.1994236E 00 I	1	SOLVED BY DIRECT METHOD

Exhibit C.

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.3000001E 00 + 0.2925553E-09 I	1	0.4829629E 00 + 0.1294095E 00 I
ROOT(2) = -0.2000000E 00 + 0.2000000E 00 I	1	0.7071067E 00 + 0.7071068E 00 I
ROOT(3) = 0.9999996E 00 + 0.1000000E 01 I	1	0.3882295E 00 + 0.1448888E 01 I
ROOT(4) = 0.9999995E 00 + -0.4528033E-08 I	1	-0.5176364E 00 + 0.1931851E 01 I
ROOT(5) = -0.1000002E 01 + -0.1000001E 01 I	1	-0.1767765E 01 + 0.1767768E 01 I
ROOT(6) = -0.3333334E 01 + 0.1599741E-06 I	1	-0.2897776E 01 + 0.7764602E 00 I
ROOT(7) = -0.1000000E 01 + 0.8811071E-08 I	1	-0.3380741E 01 + -0.9058627E 00 I
ROOT(8) = -0.1999999E 01 + -0.3000000E 01 I	1	-0.2828430E 01 + -0.2828424E 01 I
ROOT(9) = 0.2000004E 01 + 0.3921901E-06 I	1	-0.1164691E 01 + -0.4346664E 01 I
ROOT(10) = 0.2999994E 01 + 0.4597451E-06 I	1	0.1294087E 01 + -0.4829631E 01 I
ROOT(11) = 0.4144528E-07 + 0.1000000E 01 I	1	0.3889081E 01 + -0.3889092E 01 I
ROOT(12) = 0.1809455E-06 + -0.1000000E 01 I	1	0.5795551E 01 + -0.1552923E 01 I
ROOT(13) = 0.3999998E 01 + 0.4000003E 01 I	1	0.6278520E 01 + 0.1682312E 01 I
ROOT(14) = 0.2000000E 01 + -0.1000001E 01 I	1	SOLVED BY DIRECT METHOD
ROOT(15) = 0.2000000E 00 + -0.2000000E 00 I	1	SOLVED BY DIRECT METHOD

Exhibit C. Roots Are: $-1 - i$, $1 + i$, $-2 - 3i$, $2 - i$, 3 , 2 , i ,
 $-i$, $-10/3$, $.3$, -1 , 1 , $4 + 4i$, $-.2 + .2i$, $.2 - .2i$.

MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 7 OF DEGREE 15

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.3000000000000000 01 + 0.0000000000000000 00 I
P(2) = -0.1790000000000000 02 + -0.0000000000000000 00 I
P(3) = 0.2010000000000000 02 + -0.6576000000000010 02 I
P(4) = 0.1745000000000000 03 + 0.2843680000000000 03 I
P(5) = -0.7594200000000000 03 + 0.3118080000000000 03 I
P(6) = 0.8274360000000000 03 + -0.3069040000000000 04 I
P(7) = 0.1329084000000000 04 + 0.4710624000000000 04 I
P(8) = -0.5611860000000000 04 + -0.1674576000000000 04 I
P(9) = 0.7224756000000000 04 + -0.1548288000000000 04 I
P(10) = -0.2276992000000000 04 + 0.3046320000000000 04 I
P(11) = -0.1241472000000000 04 + -0.4097280000000000 04 I
P(12) = 0.5402800000000010 04 + 0.1263488000000000 04 I
P(13) = -0.6468336000000010 04 + 0.1236480000000000 04 I
P(14) = 0.1467456000000000 04 + 0.2272000000000000 02 I
P(15) = -0.1077120000000000 03 + -0.5475840000000000 03 I
P(16) = 0.3456000000000000 02 + 0.1267200000000000 03 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.3000000000000000 00 + 0.33839987517518660-16 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = 0.2000000000000000 00 + -0.2000000000000000 00 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
ROOT(3) = 0.9999999999999999 00 + 0.999999999999997D 00 I	1	0.3882284792654056D 00 + 0.1448888763117193D 01 I
ROOT(4) = -0.23890824083133820-16 + 0.1000000000000000 01 I	1	-0.5176382551966724D 00 + 0.1931851608368755D 01 I
ROOT(5) = -0.1000000000000000 01 + 0.25316292776176400-15 I	1	-0.17677671470807010 01 + 0.1767766758852015D 01 I
ROOT(6) = -0.3333333333333334D 01 + -0.21447226690877370-15 I	1	-0.2897777583074990D 01 + 0.7764567463987070D 00 I
ROOT(7) = -0.1000000000000000 01 + -0.9999999999999961D 00 I	1	-0.3380740248331229D 01 + -0.9058671940816160D 00 I
ROOT(8) = -0.2000000000000010 01 + -0.3000000000000000 01 I	1	-0.2828426607107896D 01 + -0.2828427642384390D 01 I
ROOT(9) = -0.11388167309366500-12 + -0.100000000000109D 01 I	1	-0.1164684801399899D 01 + -0.4346666459873368D 01 I
ROOT(10) = 0.1999999999999879D 01 + -0.999999999999954D 00 I	1	0.1294096345098645D 01 + -0.4829628831453027D 01 I
ROOT(11) = 0.2999999999999946D 01 + 0.19218912044391240-14 I	1	0.3889088292979509D 01 + -0.3889086300072258D 01 I
ROOT(12) = 0.200000000000122D 01 + -0.2254154879580170D-12 I	1	0.579555393512303D 01 + -0.1552912644268974D 01 I
ROOT(13) = 0.9999999999999999D 01 + 0.400000000000005D 01 I	1	0.6278517357734125D 01 + 0.1682325708247752D 01 I
ROOT(14) = 0.10000000000002080 01 + 0.22974446419373370-12 I	1	SOLVED BY DIRECT METHOD
ROOT(15) = -0.2000000000000366D 00 + 0.200000000000595D 00 I	1	SOLVED BY DIRECT METHOD

Exhibit D.

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.3000000000000000 00 + 0.337673962083244D-16 I	1	0.48296291156562790 00 + 0.1294095284438187D 00 I
ROOT(2) = 0.2000000000000000 00 + -0.2000000000000000 00 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
ROOT(3) = 0.9999999999999995D 00 + 0.1000000000000000 01 I	1	0.3882284792654056D 00 + 0.144888763117193D 01 I
ROOT(4) = 0.1153775775153668D-16 + 0.1000000000000000 01 I	1	-0.5176382551966724D 00 + 0.1931851608368755D 01 I
ROOT(5) = -0.1000000000000000 01 + -0.9935626030016848D-17 I	1	-0.1767767147080701D 01 + 0.1767766758852015D 01 I
ROOT(6) = -0.3333333333333334D 01 + -0.1919125397153664D-15 I	1	-0.2897777583074990D 01 + 0.7764567463987070D 00 I
ROOT(7) = -0.1000000000000000 01 + -0.9999999999999980 00 I	1	-0.3380740248331729D 01 + -0.9058671940816160D 00 I
ROOT(8) = -0.2000000000000000 01 + -0.3000000000000000 01 I	1	-0.2828426607107896D 01 + -0.2828427642384390D 01 I
ROOT(9) = 0.5529135233827724D-16 + -0.1000000000000000 01 I	1	-0.1164684801399899D 01 + -0.4346666459873368D 01 I
ROOT(10) = 0.2000000000000002D 01 + -0.9999999999999986D 00 I	1	0.1294096345098645D 01 + -0.4829628831453027D 01 I
ROOT(11) = 0.2999999999999996D 01 + 0.243947732911144D-14 I	1	0.3889088297979509D 01 + -0.3889086300072258D 01 I
ROOT(12) = 0.2000000000000002D 01 + -0.4102050490016657D-14 I	1	0.5795555393512303D 01 + -0.1552912644268974D 01 I
ROOT(13) = 0.4000000000000000 01 + 0.4000000000000001D 01 I	1	0.6278517357734125D 01 + 0.1682325708247752D 01 I
ROOT(14) = 0.1000000000000000 01 + 0.7682438283857070D-16 I	1	SOLVED BY DIRECT METHOD
ROOT(15) = -0.2000000000000000 00 + 0.2000000000000000 00 I	1	SOLVED BY DIRECT METHOD

Exhibit D. Roots Are: $-1 - i$, $1 + i$, $-2 - 3i$, $2 - i$, 3 , 2 , i , $-i$, $-10/3$, $.3$, -1 , 1 , $4 + 4i$, $-.2 + .2i$, $.2 - .2i$.

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 53 OF DEGREE 6

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000E 01 + 0.0000000E 00 I
P(2) = -0.7000000E 01 + -0.1050000E 02 I
P(3) = -0.2800000E 02 + 0.5800000E 02 I
P(4) = 0.1710000E 03 + 0.1500000E 01 I
P(5) = -0.7300000E 02 + -0.2510000E 03 I
P(6) = -0.2280000E 03 + 0.1040000E 03 I
P(7) = 0.7200000E 02 + 0.1040000E 03 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.10E-04
TEST FOR MULTIPLICITIES. 0.10E-01
RADIUS TO START SEARCH. 0.00E 00
RADIUS TO END SEARCH. 0.00E 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ZEROS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.9940745E 00 + 0.2009475E 01 I	2	0.4829629E 00 + 0.1294095E 00 I
ROOT(2) = 0.1814509E 01 + 0.1788034E 01 I	1	0.7071067E 00 + 0.7071068E 00 I
ROOT(3) = 0.2229694E 01 + 0.1938903E 01 I	1	0.3882295E 00 + 0.1448888E 01 I
ROOT(4) = 0.1968111E 01 + 0.2254501E 01 I	1	SOLVED BY DIRECT METHOD
ROOT(5) = -0.1000464E 01 + 0.4996119E 00 I	1	SOLVED BY DIRECT METHOD

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(2) = 0.1814509E 01 + 0.1788034E 01 I DID NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(3) = 0.2229694E 01 + 0.1938903E 01 I DID NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(4) = 0.1968111E 01 + 0.2254501E 01 I DID NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ZEROS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.9940745E 00 + 0.2009475E 01 I	2	0.4829629E 00 + 0.1294095E 00 I
ROOT(2) = 0.2049402E 01 + 0.2042129E 01 I	1	0.7071067E 00 + 0.7071068E 00 I
ROOT(3) = 0.1937237E 01 + 0.2044566E 01 I	1	0.3882295E 00 + 0.1448888E 01 I
ROOT(4) = 0.1941931E 01 + 0.2030631E 01 I	1	SOLVED BY DIRECT METHOD
ROOT(5) = -0.9999999E 00 + 0.4999999E 00 I	1	SOLVED BY DIRECT METHOD

Exhibit E. Roots Are: $2 + 2i$ (3), $1 + 2i$ (2), $-1 + .5i$ (1).

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 53 OF DEGREE 6

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000D 01 + 0.0000000000000000 00 I
P(2) = -0.7000000000000000D 01 + -0.1050000000000000 02 I
P(3) = -0.2800000000000000D 02 + 0.5800000000000000D 02 I
P(4) = 0.1710000000000000D 03 + 0.1500000000000000 01 I
P(5) = -0.7300000000000000D 02 + -0.2510000000000000 03 I
P(6) = -0.2280000000000000D 03 + 0.1040000000000000 03 I
P(7) = 0.7200000000000000D 02 + 0.1040000000000000 03 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.10D-09
TEST FOR MULTIPLICITIES. 0.10D-01
RADIUS TO START SEARCH. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
RDDT(1) = 0.9999998836125019D 00 + 0.2000000052138284D 01 I	2	0.4829629115656279D 00 + 0.1294095284438187D 00 I
RDDT(2) = 0.1996737810257486D 01 + 0.1995253821143684D 01 I	3	0.7071067553046346D 00 + 0.7071068070684595D 00 I
RDDT(3) = -0.9902131979974624D 00 + 0.5142384322923812D 00 I	1	SOLVED BY DIRECT METHOD

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(2) = 0.1996737810257486D 01 + 0.1995253821143684D 01 I DID NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.9999998836125019D 00 + 0.2000000052138284D 01 I	2	0.4829629115656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = 0.1999992907503309D 01 + 0.1999959474001689D 01 I	3	0.7071067553046346D 00 + 0.7071068070684595D 00 I
ROOT(3) = -0.999999999999998D 00 + 0.5000000000000000D 00 I	1	SOLVED BY DIRECT METHOD

Exhibit F. Roots Are: $2 + 2i$ (3), $1 + 2i$ (2), $-1 + .5i$ (1).

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 46 OF DEGREE 8

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = 0.0000000000000000 00 + -0.3000000000000000 01 I
P(3) = -0.1200000000000000 02 + -0.2000000000000000 01 I
P(4) = 0.4000000000000000 01 + 0.2800000000000000 02 I
P(5) = 0.6000000000000000 02 + 0.0000000000000000 00 I
P(6) = -0.1600000000000000 02 + -0.1160000000000000 03 I
P(7) = -0.1280000000000000 03 + 0.8000000000000000 01 I
P(8) = 0.0000000000000000 00 + 0.1440000000000000 03 I
P(9) = 0.6400000000000000 02 + -0.3200000000000000 02 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.10D-04
TEST FOR MULTIPLICITIES. 0.10D-01
RADIUS TO START SEARCH. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
RDDT(1) = 0.9999080132939515D 00 + 0.9999050266032582D 00 I	4	0.4829629115656279D 00 + 0.1294095284438187D 00 I
RDDT(2) = -0.1856970314887275D 01 + 0.2861249814184128D-01 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
RDDT(3) = -0.2048443782235651D 01 + -0.1319376429987634D 00 I	1	0.3882284792654056D 00 + 0.1448888763117193D 01 I
RDDT(4) = 0.1999808633077203D 01 + -0.9998575268225686D 00 I	1	SOLVED BY DIRECT METHOD
RDDT(5) = -0.2094026589130082D 01 + 0.1035625652664575D 00 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
RDDT(1) = 0.9998985119617400D 00 + 0.9999213371300168D 00 I	4	0.4829629115656279D 00 + 0.1294095284438187D 00 I
RDDT(2) = -0.1999961557953157D 01 + 0.7862362479129184D-05 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
RDDT(3) = -0.2000007986928054D 01 + -0.2879153403652409D-04 I	1	0.3882284792654056D 00 + 0.1448888763117193D 01 I
RDDT(4) = 0.2000000000000041D 01 + -0.999999999999982D 00 I	1	SOLVED BY DIRECT METHOD
RDDT(5) = -0.2000017849400097D 01 + 0.2356567534440272D-04 I	1	SOLVED BY DIRECT METHOD

Exhibit G. Roots Are: $1 + i$ (4), -2 (3), $2 - i$ (1).

MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 25 OF DEGREE 8

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = -0.5000000000000000 01 + -0.1150000000000000 02 I
P(3) = -0.5175000000000000 02 + 0.4300000000000000 02 I
P(4) = 0.1572500000000000 03 + 0.1446250000000000 03 I
P(5) = 0.3075000000000000 03 + -0.3475000000000000 03 I
P(6) = -0.4952500000000000 03 + -0.4948750000000000 03 I
P(7) = -0.5857500000000000 03 + 0.4247500000000000 03 I
P(8) = 0.1810000000000000 03 + 0.4420000000000000 03 I
P(9) = 0.1580000000000000 03 + 0.6000000000000000 01 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.20000422028730180 01 + 0.20000287439982940 01 I	3	0.48296291156562790 00 + 0.12940952844381870 00 I
ROOT(2) = 0.10005638069089880 01 + 0.19806389525682160 01 I	1	0.70710675530463460 00 + 0.70710680706845950 00 I
ROOT(3) = -0.10635284187498440 01 + 0.49423723791991200 00 I	1	0.38822847926540560 00 + 0.14488887631171930 01 I
ROOT(4) = -0.97221225168698240 00 + 0.56037974579731250 00 I	1	-0.51763825519667240 00 + 0.19318516083687550 01 I
ROOT(5) = 0.99943706805064100 00 + 0.20190942869286920 01 I	1	SOLVED BY DIRECT METHOD
ROOT(6) = -0.96438681314185650 00 + 0.44556354479098650 00 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.20000421758895910 01 + 0.20000287507138950 01 I	3	0.48296291156562790 00 + 0.12940952844381870 00 I
ROOT(2) = 0.10000000231652550 01 + 0.19999998057777000 01 I	1	0.70710675530463460 00 + 0.70710680706845950 00 I
ROOT(3) = -0.10000065941223140 01 + 0.49999528211260520 00 I	1	0.38822847926540560 00 + 0.14488887631171930 01 I
ROOT(4) = -0.10000009725718650 01 + 0.50000841757266510 00 I	1	-0.51763825519667240 00 + 0.19318516083687550 01 I
ROOT(5) = 0.99999991684094000 00 + 0.2000001001326590 01 I	1	SOLVED BY DIRECT METHOD
ROOT(6) = -0.99999257381295570 00 + 0.49999628183377030 00 I	1	SOLVED BY DIRECT METHOD

Exhibit H. Roots Are: $2 + 2i$ (3), $1 + 2i$ (2), $-1 + .5i$ (3).

MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 46 OF DEGREE 8

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000E 01 + 0.0000000E 00 I
P(2) = 0.0000000E 00 + -0.3000000E 01 I
P(3) = -0.1200000E 02 + -0.2000000E 01 I
P(4) = 0.4000000E 01 + 0.2800000E 02 I
P(5) = 0.6000000E 02 + 0.0000000E 00 I
P(6) = -0.1600000E 02 + -0.1160000E 03 I
P(7) = -0.1280000E 03 + 0.8000000E 01 I
P(8) = 0.0000000E 00 + 0.1440000E 03 I
P(9) = 0.6400000E 02 + -0.3200000E 02 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.10E-04
TEST FOR MULTIPLICITIES. 0.10E-01
RADIUS TO START SEARCH. 0.00E 00
RADIUS TO END SEARCH. 0.00E 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.1023019E 01 + 0.1016063E 01 I	2	0.4829629E 00 + 0.1294095E 00 I
ROOT(2) = 0.1000158E 01 + 0.9504023E 00 I	1	0.7071067E 00 + 0.7071068E 00 I
ROOT(3) = 0.9538472E 00 + 0.1017488E 01 I	1	0.3882295E 00 + 0.1448888E 01 I
ROOT(4) = -0.1973325E 01 + -0.2449018E-01 I	1	-0.5176364E 00 + 0.1931851E 01 I
ROOT(5) = -0.2033645E 01 + -0.1098356E-01 I	1	-0.1767765E 01 + 0.1767768E 01 I
ROOT(6) = 0.1999992E 01 + -0.9999957E 00 I	1	SOLVED BY DIRECT METHOD
ROOT(7) = -0.1993066E 01 + 0.3545329E-01 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.1023855E 01 + 0.1017007E 01 I	2	0.4829629E 00 + 0.1294095E 00 I
ROOT(2) = 0.1004420E 01 + 0.9757368E 00 I	1	0.7071067E 00 + 0.7071068E 00 I
ROOT(3) = 0.9693815E 00 + 0.1017069E 01 I	1	0.3882295E 00 + 0.1448888E 01 I
ROOT(4) = -0.1988549E 01 + -0.1451340E-02 I	1	-0.5176364E 00 + 0.1931851E 01 I
ROOT(5) = -0.2006490E 01 + -0.9744499E-02 I	1	-0.1767765E 01 + 0.1767768E 01 I
ROOT(6) = 0.2000000E 01 + -0.1000000E 01 I	1	SOLVED BY DIRECT METHOD
ROOT(7) = -0.2004362E 01 + 0.1135974E-01 I	1	SOLVED BY DIRECT METHOD

Exhibit I. Roots Are: $1 + i$ (4), -2 (3), $2 - i$ (1).

MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 46 OF DEGREE 8

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
P(2) = 0.0000000000000000D 00 + -0.3000000000000000D 01 I
P(3) = -0.1200000000000000D 02 + -0.2000000000000000D 01 I
P(4) = 0.4000000000000000D 01 + 0.2800000000000000D 02 I
P(5) = 0.6000000000000001D 02 + 0.0000000000000000D 00 I
P(6) = -0.1600000000000000D 02 + -0.1160000000000000D 03 I
P(7) = -0.1280000000000000D 03 + 0.8000000000000000D 01 I
P(8) = 0.0000000000000000D 00 + 0.1440000000000000D 03 I
P(9) = 0.6400000000000001D 02 + -0.3200000000000000D 02 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.10D-04
TEST FOR MULTIPLICITIES. 0.10D-01
RADIUS TO START SEARCH. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.9999385361441768D 00 + 0.9999014841207898D 00 I	3	0.4829629115656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = 0.1000184367790518D 01 + 0.1000295472219798D 01 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
ROOT(3) = -0.1995508869249228D 01 + 0.2597879064994983D-02 I	2	0.3882284792654056D 00 + 0.1448888763117193D 01 I
ROOT(4) = 0.1999994617928609D 01 + -0.1000018753040607D 01 I	1	SOLVED BY DIRECT METHOD
ROOT(5) = -0.2008976855653202D 01 + -0.5176929671550160D-02 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.1000179194957145D 01 + 0.1000267071195324D 01 I	3	0.4829629115656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = 0.1000152145562312D 01 + 0.1000254437829650D 01 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
ROOT(3) = -0.1999984369889545D 01 + -0.3932859075256354D-04 I	2	0.3882284792654056D 00 + 0.1448888763117193D 01 I
ROOT(4) = 0.2000000000000000D 01 + -0.9999999999999998D 00 I	1	SOLVED BY DIRECT METHOD
ROOT(5) = -0.2000036150881542D 01 + 0.3663652712672905D-05 I	1	SOLVED BY DIRECT METHOD

Exhibit J. Roots Are: $1 + i$ (4), -2 (3), $2 - i$ (1).

GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 25 OF DEGREE 8

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.100000DE 01 + 0.0000000E 00 I
P(2) = -0.5000000E 01 + -0.1150000E 02 I
P(3) = -0.5175000E 02 + 0.4300000E 02 I
P(4) = 0.1572500E 03 + 0.1446250E 03 I
P(5) = 0.3075000E 03 + -0.3475000E 03 I
P(6) = -0.4952500E 03 + -0.4948750E 03 I
P(7) = -0.5857500E 03 + 0.4247500E 03 I
P(8) = 0.1810000E 03 + 0.4420000E 03 I
P(9) = 0.1580000E 03 + 0.6000000E 01 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCO 0.10E-01
TEST FOR CONVERGENCE. 0.10E-04
TEST FOR MULTIPLICITIES. 0.10E 00
RADIUS TO START SEARCH. 0.00E 00
RADIUS TO END SEARCH. 0.00E 00

THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE

H(1) = 0.1000000E 01 + 0.0000000E 00 I
H(2) = -0.2000005E 01 + -0.4499880E 01 I
H(3) = -0.7000046E 01 + 0.3500016E 01 I
H(4) = 0.9995270E 00 + 0.6999649E 01 I

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ZEROS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.9994811E 00 + 0.1999989E 01 I	1	0.4829629E 00 + 0.1294095E 00 I
ROOT(2) = 0.2000473E 01 + 0.1999909E 01 I	1	SOLVED BY DIRECT METHOD
ROOT(3) = -0.9999509E 00 + 0.4999819E 00 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.9994814E 00 + 0.1999988E 01 I	2	0.4829629E 00 + 0.1294095E 00 I
ROOT(2) = 0.2000475E 01 + 0.1999910E 01 I	3	SOLVED BY DIRECT METHOD
ROOT(3) = -0.9999515E 00 + 0.4999818E 00 I	3	SOLVED BY DIRECT METHOD

Exhibit K. Roots Are: $2 + 2i$ (3), $1 + 2i$ (2), $-1 + .5i$ (3).

GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
 POLYNOMIAL NUMBER 25 OF DEGREE 8

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
 P(2) = -0.5000000000000000 01 + -0.1150000000000000 02 I
 P(3) = -0.5175000000000000 02 + 0.4300000000000000 02 I
 P(4) = 0.1572500000000000 03 + 0.1446250000000000 03 I
 P(5) = 0.3075000000000000 03 + -0.3475000000000000 03 I
 P(6) = -0.4952500000000000 03 + -0.4948750000000000 03 I
 P(7) = -0.5857500000000000 03 + 0.4247500000000000 03 I
 P(8) = 0.1810000000000000 03 + 0.4420000000000000 03 I
 P(9) = 0.1580000000000000 03 + 0.6000000000000000 01 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
 MAXIMUM NUMBER OF ITERATIONS. 200
 TEST FOR ZERO IN SUBROUTINE GCD 0.10D-02
 TEST FOR CONVERGENCE. 0.10D-09
 TEST FOR MULTIPLICITIES. 0.10D-01
 RADIUS TO START SEARCH. 0.00D 00
 RADIUS TO END SEARCH. 0.00D 00

THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE

H(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
 H(2) = -0.20000000000002730 01 + -0.45000000000002750 01 I
 H(3) = -0.70000000000009350 01 + 0.35000000000004720 01 I
 H(4) = 0.9999999999999250 00 + 0.70000000000005080 01 I

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.99999999999954140 00 + 0.1999999999997670 01 I	1	0.48296291156562790 00 + 0.12940952844381870 00 I
ROOT(2) = 0.20000000000006710 01 + 0.20000000000005310 01 I	1	SOLVED BY DIRECT METHOD
ROOT(3) = -0.99999999999994020 00 + 0.4999999999997800 00 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.99999999999954160 00 + 0.1999999999997670 01 I	2	0.48296291156562790 00 + 0.12940952844381870 00 I
ROOT(2) = 0.20000000000006710 01 + 0.20000000000005310 01 I	3	SOLVED BY DIRECT METHOD
ROOT(3) = -0.99999999999994030 00 + 0.4999999999997800 00 I	3	SOLVED BY DIRECT METHOD

Exhibit L. Roots Are: $2 + 2i$ (3), $1 + 2i$ (2), $-1 + .5i$ (3).

GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
 POLYNOMIAL NUMBER 9 OF DEGREE 12

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
 P(2) = -0.1200000000000000D 02 + -0.0000000000000000D 00 I
 P(3) = 0.7200000000000001D 02 + 0.0000000000000000D 00 I
 P(4) = -0.2800000000000000D 03 + -0.0000000000000000D 00 I
 P(5) = 0.7800000000000000D 03 + 0.0000000000000000D 00 I
 P(6) = -0.1632000000000000D 04 + -0.0000000000000000D 00 I
 P(7) = 0.2624000000000000D 04 + 0.0000000000000000D 00 I
 P(8) = -0.3264000000000000D 04 + -0.0000000000000000D 00 I
 P(9) = 0.3120000000000000D 04 + 0.0000000000000000D 00 I
 P(10) = -0.2240000000000000D 04 + -0.0000000000000000D 00 I
 P(11) = 0.1152000000000000D 04 + 0.0000000000000000D 00 I
 P(12) = -0.3840000000000000D 03 + -0.0000000000000000D 00 I
 P(13) = 0.6400000000000001D 02 + 0.0000000000000000D 00 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
 MAXIMUM NUMBER OF ITERATIONS. 200
 TEST FOR ZERO IN SUBROUTINE GCD 0.10D-02
 TEST FOR CONVERGENCE. 0.10D-09
 TEST FOR MULTIPLICITIES. 0.10D-01
 RADIUS TO START SEARCH. 0.000 00
 RADIUS TO END SEARCH. 0.000 00

THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE

H(1) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
 H(2) = -0.2000000000000015D 01 + -0.0000000000000000D 00 I
 H(3) = 0.199999999999830D 01 + 0.0000000000000000D 00 I

ZEROS OF P(X)

MULTIPLICITIES

ROOT(1) = 0.1000000000000007D 01 + 0.999999999999074D 00 I 6 SOLVED BY DIRECT METHOD
 ROOT(2) = 0.1000000000000007D 01 + -0.999999999999074D 00 I 6 SOLVED BY DIRECT METHOD

Exhibit M. Roots Are: $1 + i$ (6), $1 - i$ (6).

GREATEST COMMON DIVISOR METHOD USED WITH MULLERS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 12 OF DEGREE 15

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.4800000000000000 02 + 0.0000000000000000 00 I
P(2) = 0.2557120000000000 03 + -0.3840000000000000 03 I
P(3) = -0.7353556800000000 02 + -0.2189696000000000 04 I
P(4) = -0.3855565696000000 04 + -0.6946851456000000 04 I
P(5) = -0.1733386464800000 05 + -0.1420625972800000 05 I
P(6) = -0.4967989270400000 05 + -0.1765857464000000 05 I
P(7) = -0.1022394572130000 06 + -0.6030664232000000 04 I
P(8) = -0.1642742200560000 06 + 0.4137366230400000 05 I
P(9) = -0.2036625888420000 06 + 0.1093899227670000 06 I
P(10) = -0.1871255780010000 06 + 0.1929865440330000 06 I
P(11) = -0.1274997298590000 06 + 0.2171341227420000 06 I
P(12) = -0.2814692716800000 05 + 0.1928489727960000 06 I
P(13) = 0.1329434434800000 05 + 0.1038130226550000 06 I
P(14) = 0.3053900774700000 05 + 0.2998989141300000 05 I
P(15) = -0.1835899020000000 03 + -0.1827632160000000 03 I
P(16) = 0.2755620000000000 00 + 0.2755620000000000 00 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD. 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE

H(1) = 0.4800000000000000 02 + 0.0000000000000000 00 I
H(2) = 0.15985599999590830 03 + -0.14400000000184630 03 I
H(3) = 0.26751999998662870 03 + -0.52756800000682380 03 I
H(4) = 0.32719599997819040 03 + -0.91441600000759420 03 I
H(5) = 0.23015999960445130 02 + -0.15212520000037680 04 I
H(6) = -0.72072000047995570 02 + -0.13274279999966610 04 I
H(7) = -0.75578400005414110 03 + -0.75200400003331570 03 I
H(8) = 0.22680001335456840 01 + 0.22679999258974890 01 I

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)

MULTIPLICITIES

INITIAL APPROXIMATION

ROOT(1) = 0.30000000386885740-02 + -0.13700603048406570-09 I 1 0.48296291156562790 00 + 0.12940952844381670 00 I
ROOT(2) = 0.14653693928886120-09 + 0.10000000003025310 01 I 1 0.70710675530453460 00 + 0.70710680766845950 00 I

Exhibit N.

```

ROOT( 3) = -0.1625381061484938D-09 + 0.1499999999807065D 01 I      1      0.38822847926540560 00 + 0.14488887631171930 01 I
ROOT( 4) = 0.11404100544591550-09 + 0.30000000000539460 01 I      1      -0.5176382551966724D 00 + 0.1931851608368755D 01 I
ROOT( 5) = -0.2333333333340885D 01 + 0.4287264124730616D-11 I      1      -0.17677671470807010 01 + 0.1767766758852015D 01 I
ROOT( 6) = -0.1167273685117228D-10 + -0.1500000000006768D 01 I      1      SOLVED BY DIRECT METHOD
ROOT( 7) = -0.1000000000032260D 01 + -0.999999999855917D 00 I      1      SOLVED BY DIRECT METHOD

```

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(1) = 0.3000000038688574D-02 + -0.1370060304840657D-09 I
DID NOT CONVERGE AFTER 200 ITERATIONS
THE PRESENT APPROXIMATION IS 0.3003000000103452D-02 + -0.1371430347506515D-09 I

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.1465369084191118D-09 + 0.1000000000302531D 01 I	2	0.7071067553046346D 00 + 0.7071068070684595D 00 I
ROOT(2) = -0.1625379402673067D-09 + 0.1499999999807065D 01 I	2	0.38822847926540560 00 + 0.1448888763117193D 01 I
ROOT(3) = 0.1140410804253790D-09 + 0.3000000000053946D 01 I	3	-0.5176382551966724D 00 + 0.1931851608368755D 01 I
ROOT(4) = -0.2333333333340885D 01 + 0.4287665075729792D-11 I	1	-0.17677671470807010 01 + 0.1767766758852015D 01 I
ROOT(5) = -0.1167210227268680D-10 + -0.1500000000006767D 01 I	2	SOLVED BY DIRECT METHOD
ROOT(6) = -0.1000000000032260D 01 + -0.999999999855942D 00 I	3	SOLVED BY DIRECT METHOD

COMPILE TIME= 18.25 SEC, EXECUTION TIME= 11.78 SEC, OBJECT CODE= 35720 BYTES, ARRAY AREA= 10440 BYTES, UNUSED= 23840 BYTES

COMPILE TIME= 0.09 SEC, EXECUTION TIME= 0.00 SEC, OBJECT CODE= 35720 BYTES, ARRAY AREA= 10440 BYTES, UNUSED= 23840 BYTES

\$\$\$TOP

Exhibit N. Roots Are: $-2.33 (1)$, $.003 (2)$, $i (2)$,
 $1.5i (2)$, $-1.5i (2)$, $3i (3)$, $-1 - i (3)$.

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 25 OF DEGREE 8

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = -0.5000000000000000 01 + -0.1150000000000000 02 I
P(3) = -0.5175000000000000 02 + 0.4300000000000000 02 I
P(4) = 0.1572500000000000 03 + 0.1446250000000000 03 I
P(5) = 0.3075000000000000 03 + -0.3475000000000000 03 I
P(6) = -0.4952500000000000 03 + -0.4948750000000000 03 I
P(7) = -0.5857500000000000 03 + 0.4247500000000000 03 I
P(8) = 0.1810000000000000 03 + 0.4420000000000000 03 I
P(9) = 0.1580000000000000 03 + 0.6000000000000000 01 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.10D-04
TEST FOR MULTIPLICITIES. 0.10D-01
RADIUS TO START SEARCH. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.9999939598018138D 00 + 0.1999987697338104D 01 I	2	0.4829629115656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = -0.9842537455963138D 00 + 0.5054907465038476D 00 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
ROOT(3) = 0.2000931778142792D 01 + 0.1977492976903607D 01 I	2	0.3882284792654056D 00 + 0.1448888763117193D 01 I
ROOT(4) = -0.1010774799262452D 01 + 0.5382818238454456D 00 I	1	-0.5176382551966724D 00 + 0.1931851608368755D 01 I
ROOT(5) = 0.1998962333644498D 01 + 0.2044677936624986D 01 I	1	SOLVED BY DIRECT METHOD
ROOT(6) = -0.1005785264674943D 01 + 0.4565881445422995D 00 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.9999969810907798D 00 + 0.1999993849089791D 01 I	2	0.4829629115656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = -0.9999844151212526D 00 + 0.5000053197405852D 00 I	1	0.7071057553046346D 00 + 0.7071068070684595D 00 I
ROOT(3) = 0.2000000315627777D 01 + 0.1999940372627678D 01 I	2	0.3882284792654056D 00 + 0.1448888763117193D 01 I
ROOT(4) = -0.1000004212475928D 01 + 0.5000176446164558D 00 I	1	-0.5176382551966724D 00 + 0.1931851608368755D 01 I
ROOT(5) = 0.1999944180478862D 01 + 0.2000011983188549D 01 I	1	SOLVED BY DIRECT METHOD
ROOT(6) = -0.1000002264912354D 01 + 0.4999799648375523D 00 I	1	SOLVED BY DIRECT METHOD

Exhibit O. Roots Are: $2 + 2i$ (3), $1 + 2i$ (2), $-1 + .5i$ (3).

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
 POLYNOMIAL NUMBER 27 OF DEGREE 4

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
 P(2) = -0.4100000000000000 00 + -0.4100000000000000 00 I
 P(3) = -0.0000000000000000 00 + 0.1260700000000000 00 I
 P(4) = 0.8614099999999980-02 + -0.8614099999999980-02 I
 P(5) = -0.4414200960000000-03 + -0.0000000000000000 00 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
 MAXIMUM NUMBER OF ITERATIONS. 200
 TEST FOR CONVERGENCE. 0.100-09
 TEST FOR MULTIPLICITIES. 0.100-01
 RADIUS TO START SEARCH. 0.000 00
 RADIUS TO END SEARCH. 0.000 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
R00T(1) = 0.1019999999627143D 00 + 0.1019999999599340D 00 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
R00T(2) = 0.1039999999869989D 00 + 0.1039999999859479D 00 I	2	0.7071067553046346D 00 + 0.7071068070684595D 00 I
R00T(3) = 0.1000000000632880D 00 + 0.1000000000681702D 00 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
R00T(1) = 0.1019999999711846D 00 + 0.1019999999514636D 00 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
R00T(2) = 0.1039999999872812D 00 + 0.1039999999856656D 00 I	2	0.7071067553046346D 00 + 0.7071068070684595D 00 I
R00T(3) = 0.1010000000031534D 00 + 0.1009999999967977D 00 I	1	SOLVED BY DIRECT METHOD

Exhibit P. Roots Are: .101 + .101i, .102 + .102i, .103 + .103i, .104 + .104i.

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 27 OF DEGREE 4

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = -0.4100000000000000 00 + -0.4100000000000000 00 I
P(3) = -0.0000000000000000 00 + 0.1260700000000000 00 I
P(4) = 0.8614099999999980 -02 + -0.8614099999999980 -02 I
P(5) = -0.4414200960000000 -03 + -0.0000000000000000 00 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-17
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
RDDT(1) = 0.1019999996271430 00 + 0.1019999995993400 00 I	1	0.48296291156562790 00 + 0.12940952844381870 00 I
RDDT(2) = 0.10399999998699890 00 + 0.1039999998594790 00 I	1	0.70710675530463460 00 + 0.70710680706845950 00 I
RDDT(3) = 0.10300000003813700 00 + 0.1030000004110390 00 I	1	SOLVED BY DIRECT METHOD
RDDT(4) = 0.10100000001214980 00 + 0.1010000001301420 00 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
RDDT(1) = 0.1019999997118460 00 + 0.1019999995146360 00 I	1	0.48296291156562790 00 + 0.12940952844381870 00 I
RDDT(2) = 0.1039999998728120 00 + 0.1039999998566560 00 I	1	0.70710675530463460 00 + 0.70710680706845950 00 I
RDDT(3) = 0.1030000001823170 00 + 0.1030000003390410 00 I	1	SOLVED BY DIRECT METHOD
RDDT(4) = 0.1009999999690320 00 + 0.1009999999663820 00 I	1	SOLVED BY DIRECT METHOD

Exhibit Q. Roots Are: .101 + .101i, .102 + .102i, .103 + .103i, .104 + .104i.

MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 27 OF DEGREE 4

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = -0.4100000000000000 00 + -0.4100000000000000 00 I
P(3) = -0.0000000000000000 00 + 0.1268700000000000 00 I
P(4) = 0.8614099999999980 -02 + -0.8614099999999980 -02 I
P(5) = -0.4414200960000000 -03 + -0.0000000000000000 00 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-17
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.10399999998692660 00 + 0.10399999998745690 00 I	1	0.48296291156562790 00 + 0.12940952844381870 00 I
ROOT(2) = 0.10300000003827130 00 + 0.10300000003664080 00 I	1	0.70710675530463460 00 + 0.70710680706845950 00 I
ROOT(3) = 0.10199999996266280 00 + 0.10199999996433260 00 I	1	SOLVED BY DIRECT METHOD
ROOT(4) = 0.10100000001213920 00 + 0.10100000001156950 00 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.10399999998692950 00 + 0.10399999998746730 00 I	1	0.48296291156562790 00 + 0.12940952844381870 00 I
ROOT(2) = 0.10300000002449830 00 + 0.10300000001512210 00 I	1	0.70710675530463460 00 + 0.70710680706845950 00 I
ROOT(3) = 0.10199999998291680 00 + 0.10199999997173300 00 I	1	SOLVED BY DIRECT METHOD
ROOT(4) = 0.10100000000557760 00 + 0.10100000000557030 00 I	1	SOLVED BY DIRECT METHOD

Exhibit R. Roots Are: .101 + .101i, .102 + .102i, .103 + .103i, .104 + .104i.

MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 33 OF DEGREE 4

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = -0.4046000000000000 01 + -0.4086000000000000 01 I
P(3) = -0.1219800000000000 00 + 0.1239896200000000 02 I
P(4) = 0.8525941492000000 01 + -0.8277958252000000 01 I
P(5) = -0.4269975877160000 01 + -0.8402356471999999 01 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.10D-09
TEST FOR MULTIPLICITIES. 0.10D-17
RADIUS TO START SEARCH. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.1010000013641422D 01 + 0.1019999997158714D 01 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = 0.1010999959008221D 01 + 0.1021000008516344D 01 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
ROOT(3) = 0.1012999986290457D 01 + 0.1023000002835164D 01 I	1	SOLVED BY DIRECT METHOD
ROOT(4) = 0.1012000041059900D 01 + 0.1021999991489779D 01 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.1010999939030615D 01 + 0.1021000048991936D 01 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = 0.1011999992454635D 01 + 0.1022000010646505D 01 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
ROOT(3) = 0.1012999994777002D 01 + 0.1022999994537829D 01 I	1	SOLVED BY DIRECT METHOD
ROOT(4) = 0.1012000096743678D 01 + 0.1022000043617003D 01 I	1	SOLVED BY DIRECT METHOD

Exhibit S. Roots Are: 1.010 + 1.020i, 1.011 + 1.021i,
1.012 + 1.022i, 1.013 + 1.023i.

GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 27 OF DEGREE 4

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = -0.4100000000000000 00 + -0.4100000000000000 00 I
P(3) = -0.0000000000000000 00 + 0.1260700000000000 00 I
P(4) = 0.8614099999999980 -02 + -0.8614099999999980 -02 I
P(5) = -0.4414200960000000 -03 + -0.0000000000000000 00 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE

H(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
H(2) = -0.1025000000000000 00 + -0.1025000000000000 00 I

ZEROS OF P(X)

MULTIPLICITIES

ROOT(1) = 0.1025000000000000 00 + 0.1025000000000000 00 I

4 SOLVED BY DIRECT METHOD

Exhibit T. Roots Are: .101 + .101i, .102 + .102i,
.103 + .103i, .104 + .104i.

GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 27 OF DEGREE 4

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
P(2) = -0.4100000000000000D 00 + -0.4100000000000000D 00 I
P(3) = -0.0000000000000000D 00 + 0.1260700000000000D 00 I
P(4) = 0.861409999999998D-02 + -0.861409999999998D-02 I
P(5) = -0.4414200960000000D-03 + -0.0000000000000000D 00 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERD IN SUBROUTINE GCD 0.10D-19
TEST FOR CONVERGENCE. 0.10D-09
TEST FOR MULTIPLICITIES. 0.10D-09
RADIUS TO START SEARCH. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00

THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE

H(1) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
H(2) = -0.4100000000000000D 00 + -0.4100000000000000D 00 I
H(3) = -0.0000000000000000D 00 + 0.1260700000000000D 00 I
H(4) = 0.861409999999998D-02 + -0.861409999999998D-02 I
H(5) = -0.4414200960000000D-03 + -0.0000000000000000D 00 I

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.1019999999627143D 00 + 0.101999999959340D 00 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = 0.1039999999869989D 00 + 0.1039999999859479D 00 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
ROOT(3) = 0.10300000000381370D 00 + 0.1030000000411039D 00 I	1	SOLVED BY DIRECT METHOD
ROOT(4) = 0.1010000000121498D 00 + 0.1010000000130142D 00 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = 0.1019999999711846D 00 + 0.1019999999514636D 00 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = 0.1039999999872812D 00 + 0.1039999999856656D 00 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
ROOT(3) = 0.10300000000182317D 00 + 0.10300000000339041D 00 I	1	SOLVED BY DIRECT METHOD
ROOT(4) = 0.1009999999969032D 00 + 0.1009999999966382D 00 I	1	SOLVED BY DIRECT METHOD

Exhibit U. Roots Are: .101 + .101i, .102 + .102i, .103 + .103i, .104 + .104i.

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 33 OF DEGREE 4

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = -0.4046000000000000 01 + -0.4086000000000000 01 I
P(3) = -0.1219800000000000 00 + 0.1239896200000000 02 I
P(4) = 0.8525941492000000 01 + -0.8277958252000000 01 I
P(5) = -0.4269975877160000 01 + -0.8402356471999999D-01 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-07
TEST FOR MULTIPLICITIES. 0.100-06
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
RDDT(1) = 0.1010000020242410D 01 + 0.1020000000258103D 01 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
RDDT(2) = 0.1010999939199952D 01 + 0.1020999999178984D 01 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
RDDT(3) = 0.1012999979682521D 01 + 0.1022999999697274D 01 I	1	SOLVED BY DIRECT METHOD
RDDT(4) = 0.1012000060875117D 01 + 0.1022000000865640D 01 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
RDDT(1) = 0.1010000013301666D 01 + 0.1020000007195040D 01 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
RDDT(2) = 0.1010999960033530D 01 + 0.1020999978369735D 01 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
RDDT(3) = 0.1012999986621423D 01 + 0.1022999992757863D 01 I	1	SOLVED BY DIRECT METHOD
RDDT(4) = 0.101200004062344D 01 + 0.1021999966196982D 01 I	1	SOLVED BY DIRECT METHOD

Exhibit V. Roots Are: $1.010 + 1.020i$, $1.011 + 1.021i$,
 $1.012 + 1.022i$, $1.013 + 1.023i$.

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 37 OF DEGREE 6

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = -0.1104600000000000 02 + -0.7086000000000001 01 I
P(3) = 0.3794202000000000 02 + -0.6713896200000001 02 I
P(4) = 0.1476868749200000 02 + -0.2412407522520000 03 I
P(5) = -0.2650544690771600 03 + 0.3033533037232799 03 I
P(6) = 0.3330998887979590 03 + -0.4935380807147997 02 I
P(7) = -0.9276313939143995 02 + -0.6162818070408000 02 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.10D-09
TEST FOR MULTIPLICITIES. 0.100-17
RADIUS TO START SEARCH. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00

NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER 200 ITERATIONS.

0.4829629115656279D 00 + 0.1294095284438187D 00 I INITIAL APPROXIMATION
-0.4829629115656279D 00 + -0.1294095284438187D 00 I ALTERED APPROXIMATION

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
R00T(1) = 0.1009999396981447D 01 + 0.1019999951561502D 01 I	1	0.1294094930884686D 00 + 0.4829629210390644D 00 I
R00T(2) = 0.1011001815251888D 01 + 0.1021000145224417D 01 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
R00T(3) = 0.1011998180643081D 01 + 0.1021999853333628D 01 I	1	0.3882284792654056D 00 + 0.1448888763117193D 01 I
R00T(4) = 0.1013000607113586D 01 + 0.1023000049880457D 01 I	1	-0.5176382551966724D 00 + 0.1931851608366755D 01 I
R00T(5) = 0.3999999999999999D 01 + -0.2000000000000001D 01 I	1	SOLVED BY DIRECT METHOD
R00T(6) = 0.3000000000000001D 01 + 0.4999999999999998D 01 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
R00T(1) = 0.1009999396981447D 01 + 0.1019999951561502D 01 I	1	0.1294094930884686D 00 + 0.4829629210390644D 00 I
R00T(2) = 0.1011001815251888D 01 + 0.1021000145224417D 01 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
R00T(3) = 0.1011998062406218D 01 + 0.1021999572143372D 01 I	1	0.3882284792654056D 00 + 0.1448888763117193D 01 I
R00T(4) = 0.1013000626481528D 01 + 0.1023000085349989D 01 I	1	-0.5176382551966724D 00 + 0.1931851608368755D 01 I
R00T(5) = 0.4000000000000001D 01 + -0.2000000000000001D 01 I	1	SOLVED BY DIRECT METHOD
R00T(6) = 0.3000000000000000D 01 + 0.5000000000000001D 01 I	1	SOLVED BY DIRECT METHOD

Exhibit W. Roots Are: $1.010 + 1.020i$, $1.011 + 1.021i$, $1.012 + 1.022i$,
 $1.013 + 1.023i$, $4 - 2i$, $3 + 5i$.

GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 55 OF DEGREE 9

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = 0.1320230000000000 02 + -0.2403000000000000 00 I
P(3) = 0.5562569530000000 02 + -0.1214820634000000 02 I
P(4) = 0.4450453890000000 02 + -0.1268725824200000 03 I
P(5) = -0.3230724061600000 03 + -0.4865587937200000 03 I
P(6) = -0.1348266779280000 04 + -0.6694374018400000 03 I
P(7) = -0.2678658554260000 04 + -0.1648687451800000 03 I
P(8) = -0.2299189598500000 04 + 0.2622280045000000 03 I
P(9) = -0.4124338680000000 03 + 0.4266134510000000 03 I
P(10) = 0.1281640000000000 01 + 0.4358727000000000 02 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE

H(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
H(2) = 0.41011499942881200 01 + -0.21201499984740130 01 I
H(3) = -0.98357000157980320 01 + -0.13682899989470170 02 I
H(4) = -0.37573449887637470 02 + -0.51134499415088150 01 I
H(5) = -0.54140999720729690 02 + 0.53699499854306000 02 I
H(6) = 0.95000013416888500 00 + 0.11064999375617660 02 I

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = -0.10114999238946570 00 + 0.12014999513506210 00 I	1	0.48296291156562790 00 + 0.12940952844381870 00 I
ROOT(2) = -0.14886143908794770-03 + 0.2000000002308530 01 I	1	0.70710675530463460 00 + 0.70710680706845950 00 I
ROOT(3) = -0.2000000002648610 01 + -0.9999999993403730 00 I	1	0.38822847926540560 00 + 0.14488887631171930 01 I
ROOT(4) = 0.2999999997704550 01 + 0.9999999998985340 00 I	1	SOLVED BY DIRECT METHOD
ROOT(5) = -0.4999999999156330 01 + 0.52280390015853360-10 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = -0.10114999238946570 00 + 0.12014999513506210 00 I	2	0.48296291156562790 00 + 0.12940952844381870 00 I
ROOT(2) = -0.14886143759515140-03 + 0.2000000002308530 01 I	1	0.70710675530463460 00 + 0.70710680706845950 00 I
ROOT(3) = -0.2000000002648610 01 + -0.9999999993403740 00 I	3	0.38822847926540560 00 + 0.14488887631171930 01 I
ROOT(4) = 0.2999999997704550 01 + 0.9999999998985460 00 I	1	SOLVED BY DIRECT METHOD
ROOT(5) = -0.4999999999156340 01 + 0.52280390015853360-10 I	2	SOLVED BY DIRECT METHOD

Exhibit X. Roots Are: $-2 - i$ (3), -5 (2), $3 + i$, $2i$,
 $-0.1011 + .1201i$, $-.1012 + .1202i$.

MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 55 OF DEGREE 9

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = 0.1320230000000000 02 + -0.2403000000000000 00 I
P(3) = 0.5567569530000000 02 + -0.1214820634000000 02 I
P(4) = 0.4450453890000000 02 + -0.1269725824000000 03 I
P(5) = -0.3230724061600000 03 + -0.4865687937200000 03 I
P(6) = -0.1348266779280000 04 + -0.6694374018400000 03 I
P(7) = -0.2678658554260000 04 + -0.1648687451800000 03 I
P(8) = -0.2299188598500000 04 + 0.2622280045000000 03 I
P(9) = -0.4124338680000000 03 + 0.4266134510000000 03 I
P(10) = 0.1281640000000000 01 + 0.4358727000000000 02 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 9
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-12
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = -0.1011999999998180 00 + 0.12020000000001160 00 I	1	-0.10114999238946570 00 + 0.12014999513506210 00 I
ROOT(2) = -0.10110000000001825 00 + 0.1200999999998840 00 I	1	-0.10114999238946570 00 + 0.12014999513506210 00 I
ROOT(3) = -0.2754555940790780-15 + 0.2000000000000000 01 I	1	-0.14886143759515140-08 + 0.20000000002308530 01 I
ROOT(4) = -0.20000001246055280 01 + -0.99999996385101410 00 I	1	-0.20000000002648610 01 + -0.9999999893403740 00 I
ROOT(5) = -0.19999999537945360 01 + -0.10000001615918840 01 I	1	-0.20000000002648610 01 + -0.9999999893403740 00 I
ROOT(6) = -0.19999999215999280 01 + -0.99999987455709920 01 I	1	-0.20000000002648610 01 + -0.9999999893403740 00 I
ROOT(7) = 0.3000000000000000 01 + 0.1000000000000000 01 I	1	0.2999999997704550 01 + 0.9999999998985460 00 I
ROOT(8) = -0.49999999645753740 01 + 0.10068112174994380-06 I	1	SOLVED BY DIRECT METHOD
ROOT(9) = -0.50000000354246330 01 + -0.10068112455325690-06 I	1	SOLVED BY DIRECT METHOD

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(7) = 0.3000000000000000 01 + 0.1000000000000000 01 I
DID NOT CONVERGE AFTER 200 ITERATIONS
THE PRESENT APPROXIMATION IS 0.30029999613761900 01 + 0.10009999871253970 01 I

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) = -0.1011999999997250 00 + 0.12020000000000430 00 I	1	-0.10114999238946570 00 + 0.12014999513506210 00 I
ROOT(2) = -0.1011999999997960 00 + 0.12020000000000460 00 I	1	-0.10114999238946570 00 + 0.12014999513506210 00 I
ROOT(3) = -0.19511269617952270-15 + 0.2000000000000000 01 I	1	-0.14886143759515140-08 + 0.20000000002308530 01 I
ROOT(4) = -0.20000001253194240 01 + -0.99999996341178420 00 I	1	-0.20000000002648610 01 + -0.9999999893403740 00 I
ROOT(5) = -0.19999999559438580 01 + -0.10000001627090500 01 I	1	-0.20000000002648610 01 + -0.9999999893403740 00 I
ROOT(6) = -0.20000000110694150 01 + -0.99999982850197650 00 I	1	-0.20000000002648610 01 + -0.9999999893403740 00 I
ROOT(7) = -0.49999999641660740 01 + 0.10056936787911960-06 I	1	SOLVED BY DIRECT METHOD
ROOT(8) = -0.50000000356163850 01 + -0.10057125626638130-06 I	1	SOLVED BY DIRECT METHOD

Exhibit Y. Roots Are: $-2 - i$ (3), -5 (2), $3 + i$, $2i$,
 $-.1011 + .1201i$, $-.1012 + .1202i$.

MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 55 OF DEGREE 9

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = 0.1320230000000000 02 + -0.2403000000000000 00 I
P(3) = 0.9562569530000000 01 D2 + -0.1214820634000000 02 I
P(4) = 0.4450453890000000 02 + -0.1268725824200000 03 I
P(5) = -0.3230724061600000 03 + -0.4865587937200000 03 I
P(6) = -0.1348267792800000 04 + -0.6694374018400000 03 I
P(7) = -0.2678658554260000 04 + -0.1648687451800000 03 I
P(8) = -0.2299188598500000 04 + 0.2622280045000000 03 I
P(9) = -0.4124338680000000 03 + 0.4266134510000000 03 I
P(10) = 0.1281640000000000 01 + 0.4358727000000000 02 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-11
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
R00T(1) = -0.1011000000000203D 00 + 0.1201000000000003D 00 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
R00T(2) = -0.101199999999797D 00 + 0.120199999999998D 00 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
R00T(3) = -0.1995278819321680D -15 + 0.2000000000000000 01 I	1	0.3882284792654056D 00 + 0.144888763117193D 01 I
R00T(4) = -0.2000007154743688D 01 + -0.9999828065437014D 00 I	1	-0.5176382551966724D 00 + 0.1931851608368755D 01 I
R00T(5) = -0.499999955556945D 01 + 0.1127307532752009D -06 I	1	-0.1767767147080701D 01 + 0.1767766758852015D 01 I
R00T(6) = -0.1999981533202796D 01 + -0.1000002400471554D 01 I	1	-0.2897777583074990D 01 + 0.7764567463987070D 00 I
R00T(7) = -0.2000011312053508D 01 + -0.1000014792984741D 01 I	1	-0.3380740248331229D 01 + -0.9058671940816160D 00 I
R00T(8) = 0.3000000000000000 01 + 0.99999999999989D 00 I	1	SOLVED BY DIRECT METHOD
R00T(9) = -0.500000044443064D 01 + -0.1127307561743507D -06 I	1	SOLVED BY DIRECT METHOD

IN THE ATTEMPT TO IMPROVE ACCURACY, R00T(4) = -0.2000007154743688D 01 + -0.9999828065437014D 00 I
DID NOT CONVERGE AFTER 200 ITERATIONS
THE PRESENT APPROXIMATION IS -0.2000008660261324D 01 + -0.9999838754875147D 00 I

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
R00T(1) = -0.101199999999823D 00 + 0.1202000000000176D 00 I	1	0.4829629115656279D 00 + 0.1294095284438187D 00 I
R00T(2) = -0.1011999999999807D 00 + 0.120199999999995D 00 I	1	0.7071067553046346D 00 + 0.7071068070684595D 00 I
R00T(3) = -0.2368948183084774D -15 + 0.2000000000000000 01 I	1	0.3882284792654056D 00 + 0.144888763117193D 01 I
R00T(4) = -0.49999995678864D 01 + 0.1129904903095989D -06 I	1	-0.1767767147080701D 01 + 0.1767766758852015D 01 I
R00T(5) = -0.1999982781479067D 01 + -0.99996091562971D 00 I	1	-0.2897777583074990D 01 + 0.7764567463987070D 00 I
R00T(6) = -0.200007036875357D 01 + -0.1000015580846282D 01 I	1	-0.3380740248331229D 01 + -0.9058671940816160D 00 I
R00T(7) = 0.3000000000000000 01 + 0.1000000000000000 01 I	1	SOLVED BY DIRECT METHOD
R00T(8) = -0.500000044052519D 01 + -0.1131737422302264D -06 I	1	SOLVED BY DIRECT METHOD

Exhibit Z. Roots Are: $-2 - i$ (3), -5 (2), $3 + i$, $2i$,
 $-.1011 + .1201i$, $-.1012 + .1202i$.

BIBLIOGRAPHY

1. Beckett, Royce and James Hurt. Numerical Calculations and Algorithms. New York: McGraw Hill, 1967.
2. Conte, S. D. Elementary Numerical Analysis. New York: McGraw Hill, 1965.
3. Henrici, Peter. Elements of Numerical Analysis. New York: John Wiley and Sons, Inc., 1964.
4. Lehmer, D. H. "A Machine Method for Solving Polynomial Equations." Jour. Assoc. Comp. Mach., Vol. 8 (1961), 151-162.
5. Muller, David E. "A Method for Solving Algebraic Equations Using an Automatic Computer." Math Tables and Aids to Comp., Vol. 10 (1956), 208-215.
6. McCalla, Thomas Richard. Introduction to Numerical Methods and Fortran Programming. New York: John Wiley and Sons, Inc., 1967.
7. Barnes, Wilfred E. Introduction to Abstract Algebra. Boston: D. C. Heath and Company, 1963.
8. Markushevich, A. I. Theory of Functions of a Complex Variable. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1965.
9. Nehari, Zeev. Introduction to Complex Analysis. Boston: Allyn and Bacon, Inc., 1968.
10. Watkins, Bruce O. "Finding Zeros of a Polynomial by the Q-D Algorithm." Communications of the Assoc. of Comp. Mach., Vol. 8 (1965), 570-574.
11. Fröberg, Carl-Erik. Introduction to Numerical Analysis. Reading, Mass.: Addison-Wesley Publishing Co., Inc., 1965.
12. Henrici, Peter. "The Quotient-Difference Algorithm." Natl. Bureau of Standards Applied Mathematics Series, Vol. 49 (1958), 23-46.

APPENDIX A

SPECIAL FEATURES OF THE PROGRAMS

Several special features have been provided in each program as an aid to the user and to improve accuracy of the results. These are explained and illustrated below.¹

1. Generating Approximations

If the user does not have initial approximations available, subroutine GENAPP can systematically generate, for an N^{th} degree polynomial, N initial approximations of increasing magnitude, beginning with the magnitude specified by XSTART. If XSTART is 0., XSTART is automatically initialized to 0.5 to avoid the approximation $0. + 0.i$. The approximations are generated according to the formula:

$$X_K = (XSTART + 0.5K) (\cos \beta + i \sin \beta)$$

where

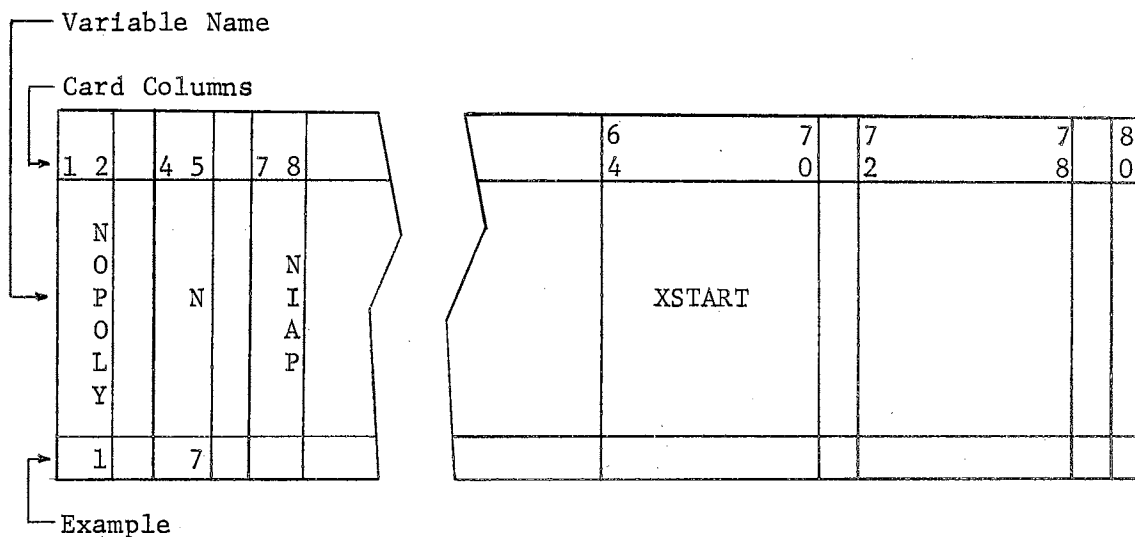
$$\beta = \frac{\pi}{12} + K \frac{\pi}{6}, \quad K = 0, 1, 2, \dots$$

To accomplish this, the user defined the number of initial approximations to be read (NIAP) on the control card to be zero (0) or these

¹These illustrations are representative of Newton's method in double precision. Control cards for single precision and other methods should be prepared accordingly.

columns (7-8) may be left blank. If XSTART is left blank, it is interpreted as 0.

For example, a portion of a control card which generates initial approximations beginning at the origin for a seventh degree polynomial is shown in example 1.



Example 1

The approximations are generated in a spiral configuration as illustrated in figure 4. Exhibit B of Chapter VII is an example of output resulting from generated approximations.

Example 2 shows a portion of a control card which generated initial approximations beginning at a magnitude of 25.0 for a sixth degree polynomial.

1	2	4	5	7	8				
N				N					
O		N		I					
P				A					
O				P					
L									
Y									
1		6							

	6	7	7	7	8
	4	0	2	8	0
	XSTART				
	2.5D + 01				

Example 2

Note that if the approximations are generated beginning at the origin, the order in which the roots are found will probably be of increasing magnitude. Roots obtained in this way are usually more accurate.

2. Altering Approximations

If an initial approximation, X_0 , does not produce convergence to a root within the maximum number of iterations, it is systematically altered a maximum of five times until convergence is possibly obtained according to the following formulas:

If the number of the alteration is odd: ($j = 1, 3$)

$$X_{j+1} = |X_0| (\cos \beta + i \sin \beta) \text{ where}$$

$$\beta = \tan^{-1} \frac{\operatorname{Im} X_0}{\operatorname{Re} X_0} + K \frac{\pi}{3}; \quad K = 1 \text{ if } j = 1, 2 \text{ if } j = 3.$$

If the number of the alteration is even: ($j = 0, 2, 4$)

$$X_{j+1} = -X_j.$$

Each altered approximation is then taken as a starting approximation. Each initial or altered approximation which does not produce convergence is printed as in Exhibit AA. If none of the six starting approximations produce convergence, the next initial approximation is taken, and the process repeated. The six approximations are spaced 60 degrees apart on a circle of radius $|X_0|$ centered at the origin as illustrated in figure 5.

3. Searching the Complex Plane

By use of initial approximations and the altering technique, any region of the complex plane in the form of an annulus centered at the origin can be searched for roots. This procedure can be accomplished in two ways.

The first way is more versatile but requires more effort on the part of the user. Specifically selected initial approximations can be used to define particular regions to be searched. For example, if the roots of a particular polynomial are known to have magnitudes between 20 and 40, an annulus of inner radius 20 and outer radius 40 could be searched by using the initial approximations $20. + i$, $23. + i$, $26. + i$, $29. + i$, $32. + i$, $35. + i$, $38. + i$, $40. + i$.

By generating initial approximations internally, the program can search an annulus centered at the origin of inner radius XSTART and outer radius XEND. Values for XSTART and XEND are supplied on the control card by the user. Example 3 shows a portion of a control card to search the above annulus of inner radius 20.0 and outer radius 40.0.

1	2	4	5	7	8
N				N	
O				I	
P		N		A	
P				P	
L					
Y					
1		7			

6	7	7	7	8
4	0	2	8	0
XSTART		XEND		
2.0D + 01		4.0D + 01		

Example 3

Note that since not less than N initial approximations can be generated at one time, the outer radius of the annulus actually searched may be greater than $XEND$ but not greater than $XEND + .5N$.

Example 4 shows a control card to search a circle of radius 15.

1	2	4	5	7	8
N				N	
O				I	
P		N		A	
P				P	
L					
Y					
2		7			

6	7	7	7	8
4	0	2	8	0
XSTART		XEND		
		1.5D + 01		

Example 4

Figure 6 shows the distribution of initial and altered approximations for an annulus of width 2 and inner radius a .

4. Improving Zeros Found

After the zeros of a polynomial are found, they are printed under the heading "Before the Attempt to Improve Accuracy." They are then used as initial approximations with Newton's (Muller's) method applied each time to the full (undeflated) polynomial. In most cases, zeros that have lost accuracy due to roundoff error in the deflation process are improved. The improved zeros are then printed under the heading "After the Attempt to Improve Accuracy." Since each root is used as an approximation to the original (undeflated) polynomial, it is possible that the root may converge to an entirely different root. As an example, see exhibit S of Chapter VII. This is especially true where several zeros are close together. Therefore, the user should check both lists of zeros to determine whether or not this has occurred. An example of improved roots is given in exhibit G of Chapter VII.

5. Solving Quadratic Polynomial

After $N-2$ roots of an N^{th} degree polynomial have been extracted, the remaining quadratic, $aX^2 + bX + c$, is solved using the quadratic formula

$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

for the two remaining roots. These are indicated by the words "Solved By Direct Method" in the initial approximation column. If only a polynomial of degree 1 is to be solved, the solution is found directly as $(X - C) = 0$ implies $X = C$.

Note that if several roots are known to the user, they may be "divided out" of the original polynomial by using this procedure.

Example 6 indicates that 2 initial approximations are supplied by the user to a 7th degree polynomial. After these approximations are used the circle or radius 15 will be searched for the remaining roots.

1	2		4	5		7	8	
N			N			N		
P						A		
O						P		
L								
Y								
1			7			2		

	6		7		7		7		8
	4		0		2		8		0
	XSTART				XEND				
							1.5D + 01		

Example 6

By defining XSTART between 0. and 15. an annulus instead of the circle will be searched (exhibit CC).

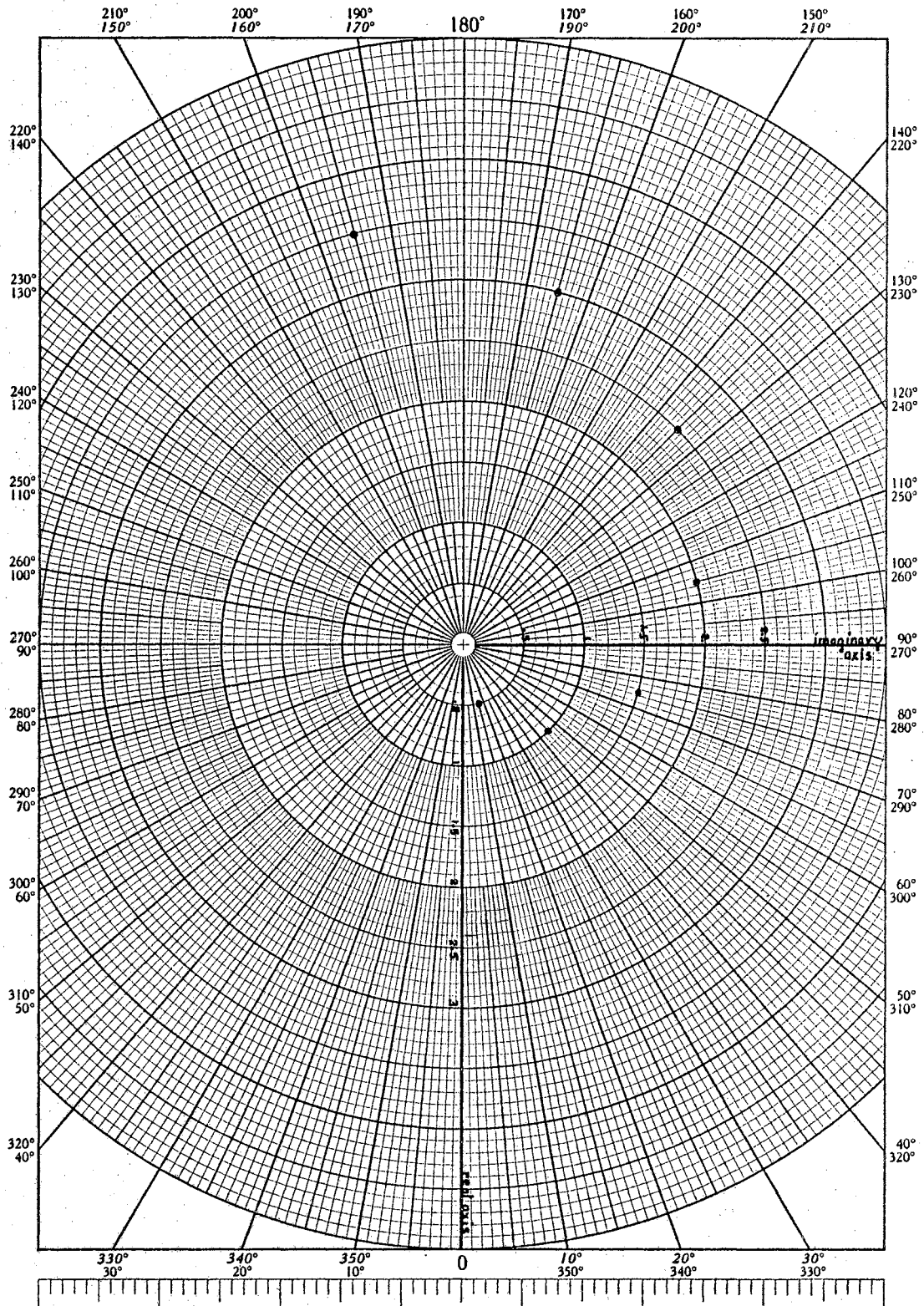


Figure 4. Generating Initial Approximations

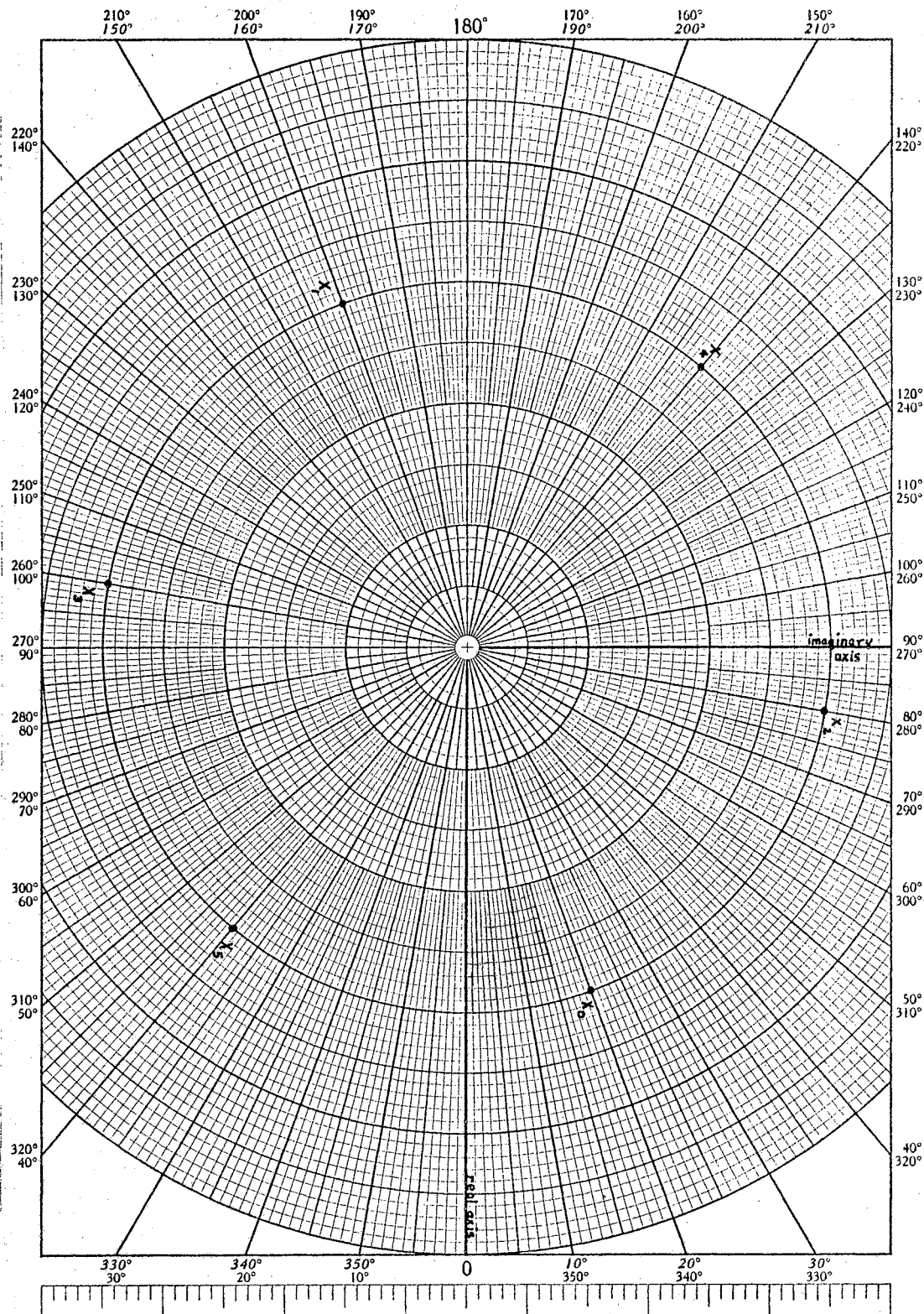


Figure 5. Altering Approximations

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 2 OF DEGREE 3

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
P(2) = 0.2000000000000000D 01 + 0.0000000000000000D 00 I
P(3) = -0.1000000000000000D 01 + -0.0000000000000000D 00 I
P(4) = -0.2000000000000000D 01 + -0.0000000000000000D 00 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 3
TEST FOR CONVERGENCE. 0.10D-03
TEST FOR MULTIPLICITIES. 0.10D-01
RADIUS TO START SEARCH. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00

NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER 3 ITERATIONS.

0.4829629115656279D 00 + 0.1294095284438187D 00 I	INITIAL APPROXIMATION
-0.4829629115656279D 00 + -0.1294095284438187D 00 I	ALTERED APPROXIMATION
0.1294094930884686D 00 + 0.4829629210390644D 00 I	ALTERED APPROXIMATION
-0.1294094930884686D 00 + -0.4829629210390644D 00 I	ALTERED APPROXIMATION
-0.3535534294161402D 00 + 0.3535533517704030D 00 I	ALTERED APPROXIMATION
0.3535534294161402D 00 + -0.3535533517704030D 00 I	ALTERED APPROXIMATION
0.7071067553046346D 00 + 0.7071068070684595D 00 I	INITIAL APPROXIMATION
-0.7071067553046346D 00 + -0.7071068070684595D 00 I	ALTERED APPROXIMATION
-0.2588191275983359D 00 + 0.9659258041843774D 00 I	ALTERED APPROXIMATION
0.2588191275983359D 00 + -0.9659258041843774D 00 I	ALTERED APPROXIMATION
-0.9659258610249968D 00 + 0.2588189154662357D 00 I	ALTERED APPROXIMATION
0.9659258610249968D 00 + -0.2588189154662357D 00 I	ALTERED APPROXIMATION
0.3882284792654056D 00 + 0.1448888763117193D 01 I	INITIAL APPROXIMATION
-0.3882284792654056D 00 + -0.1448888763117193D 01 I	ALTERED APPROXIMATION
-0.1060660288248421D 01 + 0.1060660055311209D 01 I	ALTERED APPROXIMATION
0.1060660288248421D 01 + -0.1060660055311209D 01 I	ALTERED APPROXIMATION
-0.1448888677856240D 01 + -0.3882287974635502D 00 I	ALTERED APPROXIMATION
0.1448888677856240D 01 + 0.3882287974635502D 00 I	ALTERED APPROXIMATION

COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

D(1) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
D(2) = 0.2000000000000000D 01 + 0.0000000000000000D 00 I
D(3) = -0.1000000000000000D 01 + -0.0000000000000000D 00 I
D(4) = -0.2000000000000000D 01 + -0.0000000000000000D 00 I

Exhibit AA.

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 1 OF DEGREE 7

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = -0.1000000000000000 01 + 0.1100000000000000 02 I
P(3) = -0.5900000000000000 02 + -0.2900000000000000 02 I
P(4) = 0.1950000000000000 03 + -0.1690000000000000 03 I
P(5) = 0.7000000000000000 02 + 0.7230000000000000 03 I
P(6) = -0.1624000000000000 04 + -0.6960000000000000 03 I
P(7) = 0.1922000000000000 04 + -0.1832000000000000 04 I
P(8) = 0.1596000000000000 04 + 0.1692000000000000 04 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 3
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
RDDT(1) = -0.2999999999999970 01 + -0.3000000000000000 01 I	1	-0.3500000000000000 01 + -0.3500000000000000 01 I
RDDT(2) = 0.2000000000000000 01 + 0.2000000000000000 01 I	1	0.2500000000000000 01 + 0.2500000000000000 01 I
RDDT(3) = -0.9999999999999960 00 + -0.3999999999999960 01 I	1	-0.1500000000000000 01 + -0.4500000000000000 01 I

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
RDDT(1) = -0.2999999999999960 01 + -0.3000000000000000 01 I	1	-0.3500000000000000 01 + -0.3500000000000000 01 I
RDDT(2) = 0.2000000000000000 01 + 0.2000000000000000 01 I	1	0.2500000000000000 01 + 0.2500000000000000 01 I
RDDT(3) = -0.9999999999999970 00 + -0.3999999999999960 01 I	1	-0.1500000000000000 01 + -0.4500000000000000 01 I

COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

D(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
D(2) = -0.2999999999999930 01 + 0.6000000000000000 01 I
D(3) = -0.2000000000000000 02 + -0.1899999999999990 02 I
D(4) = 0.4100000000000000 02 + -0.2200000000000000 02 I
D(5) = 0.2300000000000000 02 + 0.4100000000000000 02 I

Exhibit BB. Roots Are: $-1 - 4i$, $-2 - 3i$, $-3 - 3i$, $-1 - i$, $2 + 2i$, $4 - i$, $2 - i$.

NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 1 OF DEGREE 7

THE COEFFICIENTS OF P(X) ARE

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
P(2) = -0.1000000000000000 01 + 0.1100000000000000 02 I
P(3) = -0.5900000000000000 02 + -0.2900000000000000 02 I
P(4) = 0.1950000000000000 03 + -0.1690000000000000 03 I
P(5) = 0.7000000000000000 02 + 0.7230000000000000 03 I
P(6) = -0.1624000000000000 04 + -0.6960000000000000 03 I
P(7) = 0.1922000000000000 04 + -0.1832000000000000 04 I
P(8) = 0.1596000000000000 04 + 0.1692000000000000 04 I

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 2
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.100-01
RADIUS TO START SEARCH. 0.700 01
RADIUS TO END SEARCH. 0.150 02

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
RROOT(1) = -0.2999999999999970 01 + -0.3000000000000002 01 I	1	-0.3500000000000000 01 + -0.3500000000000000 01 I
RROOT(2) = 0.2000000000000000 01 + 0.2000000000000000 01 I	1	0.2500000000000000 01 + 0.2500000000000000 01 I
RROOT(3) = 0.4000000000000000 01 + -0.1000000000000000 01 I	1	0.6761480761918791 01 + 0.1811733398213462 01 I
RROOT(4) = 0.1999999999999970 01 + -0.9999999999999760 00 I	1	0.5303300664784760 01 + 0.5303301053013447 01 I
RROOT(5) = -0.9999999999999980 00 + -0.1000000000000004 01 I	1	0.2070551889415497 01 + 0.7727406736625032 01 I
RROOT(6) = -0.9999999999999760 00 + -0.3999999999999950 01 I	1	SOLVED BY DIRECT METHOD
RROOT(7) = -0.2000000000000004 01 + -0.3000000000000001 01 I	1	SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)	MULTIPLICITIES	INITIAL APPROXIMATION
RROOT(1) = -0.2999999999999960 01 + -0.3000000000000001 01 I	1	-0.3500000000000000 01 + -0.3500000000000000 01 I
RROOT(2) = 0.2000000000000000 01 + 0.2000000000000000 01 I	1	0.2500000000000000 01 + 0.2500000000000000 01 I
RROOT(3) = 0.4000000000000000 01 + -0.1000000000000000 01 I	1	0.6761480761918791 01 + 0.1811733398213462 01 I
RROOT(4) = 0.2000000000000000 01 + -0.9999999999999980 00 I	1	0.5303300664784760 01 + 0.5303301053013447 01 I
RROOT(5) = -0.9999999999999970 00 + -0.1000000000000000 01 I	1	0.2070551889415497 01 + 0.7727406736625032 01 I
RROOT(6) = -0.9999999999999820 00 + -0.4000000000000000 01 I	1	SOLVED BY DIRECT METHOD
RROOT(7) = -0.2000000000000003 01 + -0.3000000000000007 01 I	1	SOLVED BY DIRECT METHOD

COMPILE TIME= 14.39 SEC, EXECUTION TIME= 103.77 SEC, OBJECT CODE= 1880 BYTES, ARRAY AREA= 3400 BYTES, UNUSED= 47720 BYTES

Exhibit CC. Roots Are: $-1 - 4i$, $-2 - 3i$, $-3 - 3i$, $-1 - i$, $2 + 2i$, $4 - i$, $2 - i$.

APPENDIX B

NEWTON'S METHOD

1. Use of the Programs

Two programs using Newton's method are presented here. The first is the single precision program. The second program is in double precision and is designed to perform double precision complex arithmetic. These programs are written for use on any computer using FORTRAN IV language. They have been tested on the IBM S/360 mod. 50 computer which has a 32 bit word. However, it may be necessary to change the system functions as described below. The single precision program may be changed to double precision as described below.

After selecting the desired program, the input data should be prepared as described in section 2.

Each program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree N , where $N > 25$, certain array dimensions must be changed. These are listed in Table I for the main program and subprograms in both single precision and double precision.

TABLE I
PROGRAM CHANGES FOR SOLVING POLYNOMIALS
OF DEGREE GREATER THAN 25
BY NEWTON'S METHOD

Single PrecisionDouble Precision

Main Program

A(N+1)	RA(N+1), VA(N+1)
B(N+1)	RB(N+1), VB(N+1)
C(N+1)	RC(N+1), VC(N+1)
D(N+1)	RD(N+1), VD(N+1)
COEF(N+1)	RCOEF(N+1), VCOEF(N+1)
MULT(N)	MULT(N)
XZERO(N)	RXZERO(N), VXZERO(N)
X(N)	RX(N), VX(N)
XINIT(N)	RXINIT(N), VXINIT(N)

Subroutine HORNER

A(N+1)	RA(N+1), VA(N+1)
B(N+1)	RB(N+1), VB(N+1)
C(N+1)	RC(N+1), VC(N+1)

Subroutine BETTER

XZERO(N)	RXZERO(N), VXZERO(N)
X(N)	RX(N), VX(N)
A(N+1)	RA(N+1), VA(N+1)
COEF(N+1)	RCOEF(N+1), VCOEF(N+1)
C(N+1)	RC(N+1), VC(N+1)
B(N+1)	RB(N+1), VB(N+1)

Subroutine GENAPP

APP(N)	APPR(N), APPI(N)
--------	------------------

Subroutine QUAD

A(N+1)	UA(N+1), VA(N+1)
ROOT(N)	UROOT(N), VROOT(N)
MULTI(N)	MULTI(N)

Certain computers may require that the following system functions in the single precision and double precision programs be changed: A "c"

denotes a complex number and an "r" denotes a real number.

TABLE II
SYSTEM FUNCTIONS USED IN NEWTON'S METHOD

<u>Single Precision</u>		<u>Double Precision</u>
CABS(c)	- obtain absolute value -	DABS(r)
COS(r)	- obtain cosine of angle -	DCOS(r)
SIN(r)	- obtain sine of angle -	DSIN(r)
CMPLX(r ₁ ,r ₂)	- express two real numbers in complex form	
AIMAG(c)	- obtain imaginary part	
REAL(c)	- obtain real part	
ATAN2(r ₁ ,r ₂)	- arctangent of r ₁ /r ₂ -	DATAN2(r ₁ ,r ₂)
CSQRT(c)	- square root -	DSQRT(r)

When used on the IBM S/360 with the WATFOR compiler for FORTRAN IV, the system functions in Table II-A must be typed in a declaration statement. These also appear in the program listing. For use without the WATFOR compiler or on other computers, these system functions might have to be removed. A "c" denotes a complex number and an "r" denotes a real number.

The single precision program may be converted to double precision for use on machines equipped to perform double precision complex arithmetic provided the following changes or their equivalent are made and the system functions of Table III are used and typed in a declaration statement where necessary. The changes presented below are those

required for the IBM S/360. A "c" denotes a complex number and an "r" denotes a real number. The format statements should be changed from E-type to D-type.

In the main program and each subprogram change COMPLEX c_1, c_2, \dots to COMPLEX*16 c_1, c_2, \dots and add IMPLICIT REAL*8(A-H,O-Z).

TABLE II-A

SYSTEM FUNCTIONS IN NEWTON'S METHOD TO BE TYPED
WHEN THE WATFOR COMPILER IS USED

Single PrecisionDouble Precision

Main Program and Subroutines
NEWTON, CHECK, and BETTER

square root - DSQRT(r)

Subroutine GENAPP

CMPLX(r_1, r_2) - express in complex form
cosine of angle - DCOS(r)
sine of angle - DSIN(r)

Subroutine ALTER

CMPLX(r_1, r_2) - express in complex form
cosine of angle - DCOS(r)
sine of angle - DSIN(r)
arctangent of r_1/r_2 - DATAN2(r_1, r_2)
square root - DSQRT(r)

Subroutine QUAD

CSQRT(c) - square root - DSQRT(r)

Subroutine COMSQT

absolute value - DABS(r)
square root - DSQRT(r)

TABLE III

SYSTEM FUNCTIONS FOR CONVERTING SINGLE PRECISION
 NEWTON'S METHOD TO DOUBLE PRECISION

<u>Single Precision</u>	changed to	<u>Double Precision</u>
	Main Program and Subroutines NEWTON, CHECK, BETTER, ALTER, and QUAD	
CABS(c)	- absolute value -	CDABS(c)
	Subroutine GENAPP	
COS(r)	- cosine of angle -	DCOS(r)
SIN(r)	- sine of angle -	DSIN(r)
CMPLX(r ₁ ,r ₂)	- express in complex form -	DCMPLX(r ₁ ,r ₂)
	Subroutine ALTER	
		Add COMPLEX*8 SXOLD
CMPLX(r ₁ ,r ₂)	- express in complex form -	DCMPLX(r ₁ ,r ₂)
AIMAG(c)	- obtain imaginary part -	AIMAG(c) (single precision)
REAL(c)	- obtain real part -	REAL(c) (single precision)
ATAN2(r ₁ ,r ₂)	- arctangent of r ₁ /r ₂ -	DATAN2(r ₁ ,r ₂)
COS(r)	- cosine of angle -	DCOS(r)
SIN(r)	- sine of angle -	DSIN(r)
	Subroutine QUAD	
CSQRT(c)	- square root -	CDSQRT(c)

2. Input Data for Newton's Method

The input data for Newton's method is grouped into polynomial data sets. Each polynomial data set consists of the data for one and only one polynomial. As many polynomials as the user desires may be solved by placing the polynomial data sets one behind the other. Each polynomial data set consists of three kinds of information placed in the following order:

1. Control information
2. Coefficients of the polynomial
3. Initial approximations. These may be omitted as described in Appendix A, § 1.

An end card follows the entire collection of data sets. It indicates that there is no more data to follow and terminates execution of the program. This information is displayed in Figure 7 and described below. For single precision data, the E-type specification should be used, while for double precision data, the D-type specification should be used. All data should be right justified. The recommendations given in Table IV are those found to give best results on the IBM S/360 mod. 50 computer which has a 32 bit word. The entry in parentheses is the double precision equivalent.

Control Information

The control card is the first card of the polynomial data set and contains the information given in Table IV. See Figure 8.

TABLE IV
CONTROL DATA FOR NEWTON'S METHOD

<u>Variable Name</u>	<u>Card Columns</u>	<u>Description</u>
NOPOLY	c.c. 1-2	Number of the polynomial. Integer. Right justified.
N	c.c. 4-5	Degree of the polynomial. Integer. Right justified.

TABLE IV (Continued)

<u>Variable Name</u>	<u>Card Columns</u>	<u>Description</u>
NIAP	c.c. 7-8	Number of initial approximations to be read. Integer. If no approximations are given, this should be left blank.
MAX	c.c. 19-21	Maximum number of iterations. Integer. Right justified. 200 is recommended.
EPSCNV	c.c. 30-35	Convergence requirement. Real. Right justify. 1.E-05 (1.D-10) is recommended.
EPSQ	c.c. 37-42	Tolerance check for zero (0) in subroutine QUAD. Real. Right justify. 1.E-10 (1.D-20) is recommended.
EPSMUL	c.c. 44-49	Multiplicity requirement. Real. Right justify. 1.E-01 (1.D-02) is recommended.
XSTART	c.c. 64-70	Magnitude at which to begin generating initial approximations. Real. Right justify. This is a special feature of the program and may be omitted.
XEND	c.c. 72-78	Magnitude at which to end the generating of initial approximations. Real. Right justify. This is a special feature of the program and may be omitted.
KCHECK	c.c. 80	This should be left blank.

Coefficients of the Polynomial

The coefficient cards follow the control card. For an N^{th} degree polynomial, $N+1$ coefficients must be entered one per card. The coefficient of the highest degree term is entered first. For example, if the polynomial $X^5 + 3X^4 + 2X + 5$ were to be solved, the order in which the coefficients would be entered is: 1, 3, 0, 0, 2, 5. Each real or complex coefficient is entered, one per card, as described in Table V and illustrated in Figure 9.

TABLE V
COEFFICIENT DATA FOR NEWTON'S METHOD

<u>Variable Name</u>	<u>Card Columns</u>	<u>Description</u>
A (RA)	c.c. 1-30	Real part of complex coefficient. Real. Right justify. If none, leave blank or enter 0.0E00 (0.0D00).
A (VA)	c.c. 31-60	Imaginary part of complex number. Real. Right justify. If none, leave blank or enter 0.0E00 (0.0D00).

Initial Approximations

The initial approximation cards follow the set of coefficient cards. The number of initial approximations read must be the number

specified on the control card and are entered, one per card, as given in Table VI and illustrated in Figure 10.

TABLE VI
INITIAL APPROXIMATION DATA FOR NEWTON'S METHOD

<u>Variable Name</u>	<u>Card Columns</u>	<u>Description</u>
XZERO (RXZERO)	c.c. 1-30	Real part of complex number. Real. Right justify. If none, leave blank or enter 0.0E00 (0.0D00).
XZERO (VXZERO)	c.c. 31-60	Imaginary part of complex number. Real. Right justify. If none, leave blank or enter 0.0E00 (0.0D00).

End Card

The end card is the last card of the input data to the program. It indicates that there is no more data to be read. When this card is read, program execution is terminated. This card is described in Table VII and illustrated in Figure 11.

TABLE VII
DATA TO END EXECUTION OF NEWTON'S METHOD

<u>Variable Name</u>	<u>Card Columns</u>	<u>Description</u>
KCHECK	c.c. 80	Must contain the number 1. Integer.

3. Variables Used in Newton's Method

The definitions at the major variables used in Newton's method are given in Table VIII. The symbols used to indicate type are:

- R - real variable
- I - integer variable
- C - complex variable
- L - logical variable
- A - alphanumeric variable

When two variables are listed, the one on the left is the real part of the corresponding single precision variable; the one on the right is the imaginary part. The symbols used to indicate disposition are:

- E - entered
- R - returned
- ECR - entered, changed, and returned
- C - variable in common

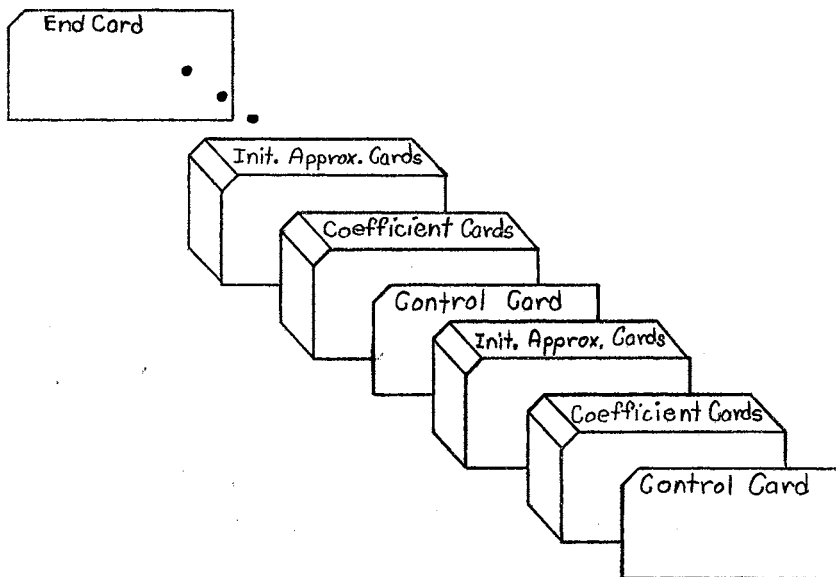


Figure 7. Sequence of Input Data for Newton's Method

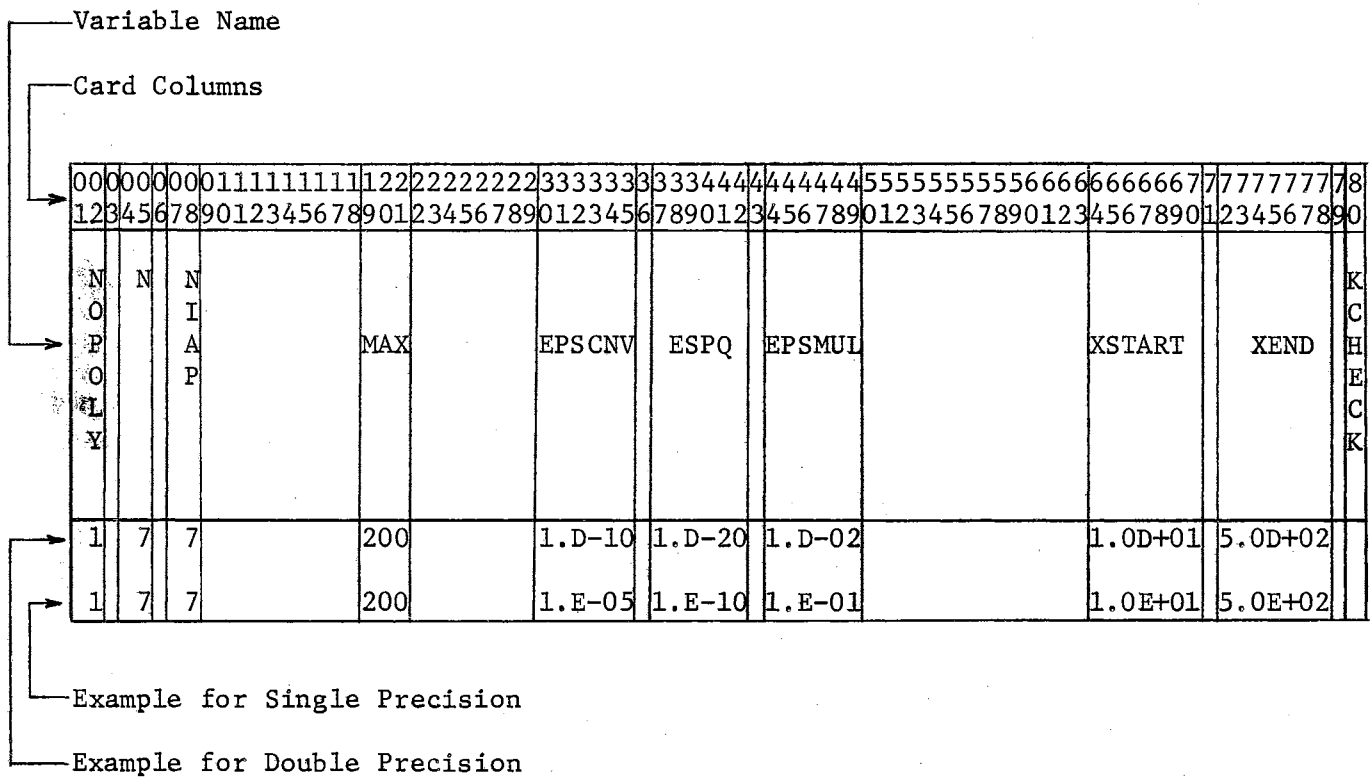


Figure 8. Control Card for Newton's Method

0000000001111111112222222222333333333444444444455555555556666666667777777778		
1234567890123456789012345678901234567890123456789012345678901234567890		
A (RA)	A (VA)	
0.621735D+01	-0.132714D-02	
0.621735E+01	-0.132714E-02	

Figure 9. Coefficient Card for Newton's Method

0000000001111111112222222222333333333344444444445555555555666666666677777777778	1234567890123456789012345678901234567890123456789012345678901234567890
XZERO (RXZERO)	XZERO (VXZERO)
0.15D+01	-0.25D-00
0.15E+01	-0.25E-00

Figure 10. Initial Approximation Card for Newton's Method

0000000001111111112222222222333333333344444444445555555555666666666677777777778	1234567890123456789012345678901234567890123456789012345678901234567890
K C H E C K 1 1	

Figure 11. End Card for Newton's Method

TABLE VIII

VARIABLES USED IN NEWTON'S METHOD

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition</u> <u>of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
Main Program					
NOPOLY	I	NOPOLY	I		Number of the polynomial
N	I	N	I		Degree of the polynomial
NIAP	I	NIAP	I		Number of initial approximations to be read
MAX	I	MAX	I		Maximum number of iterations to be performed
EPSCNV	R	EPSCNV	R		Tolerance check for convergence
EPSMUL	R	EPSMUL	R		Tolerance check for multiplicities
EPSQ	R	EPSQ	R		Tolerance check for zero in subroutine QUAD
XSTART	R	XSTART	R		Magnitude from which to begin the search for zeros
XEND	R	XEND	R		Magnitude to end the search for zeros
KCHECK	I	KCHECK	I		Program Control. When KCHECK = 1, program will terminate execution.
NA	I	NA	I		Number of coefficients of original polynomial
A	C	RA,VA	R		Array containing the coefficients of original polynomial P(X)
NDEF	I	NDEF	I		Degree of current deflated polynomial
L	I	L	I		Counter for number of initial approximations used
ITER	I	ITER	I		Counter for number of iterations
NROOT	I	NROOT	I		Counter for number of roots found (counting multiplicities)
IALTER	I	IALTER	I		Counter for number of alterations of each initial approximation
ITIME	I	ITIME	I		Program control
K	I	K	I		Counter for number of distinct roots found
ND	I	ND	I		Program control & number of coefficient of deflated polynomial for which no zeros were found

TABLE VIII (Continued)

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
XO	C	RXO,VXO	R		Current approximation (X_n) to root
COEF	C	RCOEF,VCOEF	R		Working array containing coefficients of current deflated polynomial
DPX	C	RDPX, VDPX	R		Derivative of P(X) at some value X
PX	C	RPX,VPX	R		Value of P(X) at some point X
XZERO	C	RXZERO, VXZERO	R		Array containing the initial approximations
XNEW	C	RXNEW,VXNEW	R		New approximation (X_{n+1}) obtained from old approximation (X_n) by Newton's Algorithm
KANS	I	KANS	I		KANS = 1 implies convergence, KANS = 0 implies no convergence
MULT	I	MULT	I		Array containing the number of multiplicities of each root
X	C	RX,VX	R		Array containing the zeros of P(X)
XINIT	C	RXINIT, VXINIT	R		Array containing the initial or altered approximations which produced convergence to each root
NUM	I	NUM	I		Number of coefficients of current deflated polynomial
B	C	RB,VB	R		Array containing the coefficients of newly deflated polynomial
IROOT	I	IROOT	I		Number of distinct roots found by Newton's method, i.e. not solved for directly by subroutine QUAD
D	C	RD,VD	R		Array containing the coefficients of deflated polynomial for which no zeros were found
I01	I	I01	I		Unit number of input device
I02	I	I02	I		Unit number of output device
C	C	RC,VC	R		Array containing sequence of values leading to the derivative
EPSCHK	R	EPSCHK	R		Current tolerance for checking convergence or multiplicity

TABLE VIII (Continued)

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition</u> <u>of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
Subroutine HORNER					
A	C	RA,VA	R	E	Array of coefficients of polynomial
B	C	RB,VB	R	R	Array of coefficients of deflated polynomial
NDEF	I	NDEF	I	E	Degree of polynomial
NUM	I	NUM	I		Number of coefficients of polynomial
XO	C	RXO,VXO	R	E	Point (X_n) at which to evaluate the polynomial and its derivative. Also current approximation (X_{n+1}) used to deflate the polynomial
PX	C	RPX,VPX	R	R	Value of polynomial at X_n
DPX	C	RD PX,VD PX	R	R	Value of the derivative of polynomial at X_n
C	C	RC,VC	R	R	Array of containing sequence of values leading to the derivative
Subroutine NEWTON					
PX	C	RPX,VPX	R	E	Value of polynomial at X_n
DPX	C	RD PX,VD PX	R	E	Derivative of polynomial at X_n
XO	C	RXO,VXO	R	E	Current approximation (X_n) to root
XNEW	C	RXNEW,VXNEW	R	R	New approximation (X_{n+1}) to root
Subroutine CHECK					
EPSLON	R	EPS	R	C	Tolerance for convergence or multiplicity check
PX	C	RPX,VPX	R	E	Value of $P(X)$ at X_n
DPX	C	RD PX,VD PX	R	E	Derivative of $P(X)$ at X_n
XO	C	RXO,VXO	R	E	Current approximations (X_{n+1}) to root
IO2	I	IO2	I	C	Unit number of output device
KANS	I	KANS	I	R	KANS = 1 implies convergence, KANS = 0 implies no convergence

TABLE VIII (Continued)

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
Subroutine BETTER					
I02	I	I02	I	C	Unit number of output device
XZERO	C	RXZERO, VXZERO	R	E	Array of approximations
X	C	RX,VX	R	ECR	Array of roots
A	C	RA,VA	R	E	Coefficients of original (undeflated) polynomial, P(X)
COEF	C	RCOEF,VCOEF	R	E	Working array for coefficients of polynomial
NA	I	NA	I	E	Number of coefficients of original polynomial
X0	C	RX0,VX0	R		Current approximation (X_n) to root
DPX	C	RDPX,VDPX	R		Derivative of P(X) at X_n
PX	C	RPX,VPX	R		Value of P(X) at X_n
KANS	I	KANS	I		KANS = 1 implies convergence; KANS = 0 implies no convergence
ITER	I	ITER	I		Counter for number of iterations
XNEW	C	RXNEW,VXNEW	R		New approximation (X_{n+1}) to root
NN	I	NN	I		Degree of polynomial
C	C	RC,VC	R	E	Array containing the sequence of values leading to the derivative
K	I	K	I	E	Number of distinct roots of P(X) found
N	I	N	I	E	Degree of polynomial P(X)
B	C	RB,VB	R	E	Array of coefficients of deflated polynomial
MAX	I	MAX	I	C	Maximum number of iterations permitted
EPSCHK	R	EPS	R	C	Tolerance for checking convergence
Subroutine GENAPP					
APP	C	APPR,APPI	R	R	Array containing initial approximations
NAPP	I	NAPP	I	E	Number of initial approximations to be generated

TABLE VIII (Continued)

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
XSTART	R	XSTART	R	ECR	Magnitude at which to begin generating approximations; also magnitude of the approximation being generated
BETA	R	BETA	R		Argument of the complex approximation being generated
U	R	APPR(I)	R		Real part of complex approximation
V	R	APPI(I)	R		Imaginary part. of complex approximation
Subroutine ALTER					
XOLD	C	XOLDR, XOLDI	R	ECR	Old approximation to be altered to new approximation
NALTER	I	NALTER	I	ECR	Number of alterations performed on an initial approximation
ITIME	I	ITIME	I	E	Program control
MAX	I	MAX	I	C	Maximum number of iterations permitted
Y	R	XOLDI	R		Imaginary part of original initial approximation (unaltered)
X	R	XOLDR	R		Real part of original unaltered initial approximation
R	R	R	R		Magnitude of original unaltered initial approximation
BETA	R	BETA	R		Argument of new approximation
XOLDR	R	XOLDR	R		Real part of new approximation
XOLDI	R	XOLDI	R		Imaginary part of new approximation
I02	I	I02	I	C	Unit number of output device
Subroutine QUAD					
A	C	UA, VA	R	E	Coefficients of polynomial to be solved
NA	I	NA	I	E	Degree of polynomial
ROOT	C	UROOT, VROOT	R	ECR	Array of roots of P(X) (original polynomial)
NROOT	I	NROOT	I	ECR	Number of distinct roots of P(X) (the original polynomial)

TABLE VIII (Continued)

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
MULTI	I	MULTI	I	ECR	Array containing multiplicities of each root
EPST	R	EPST	R	E	Tolerance check for the number zero
DISC	C	UDISC,VDISC	R		Value of the discriminate ($b^2 - 4ac$) of Quadratic
Subroutine COMSQT					
		UX,VX	R	E	Complex number for which the square root is desired
		UY,VY	R	R	Square root of the complex number

4. Description of Program Output

The output from Newton's method programs consist of the following information.

The number and degree of the polynomial are printed in the heading (exhibit B, Chapter VII).

The coefficients are printed under the heading "THE COEFFICIENTS OF P(X) ARE." The coefficient of the highest degree term is listed first (exhibit B, Chapter VII).

As an aid to ensure the control information is correct, the number of initial approximations given, maximum number of iterations, test for convergence, test for multiplicities, radius to start search, and radius to end search are printed as read from the control card (exhibit B, Chapter VII).

The zeros found before and after the attempt to improve accuracy are printed. See Appendix A, § 4 for further explanation (exhibit B, Chapter VII).

If not all zeros of the polynomial are found, the coefficients of the remaining unsolved polynomial will be printed, with coefficient of highest degree term first, under the heading "COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND." See Appendix A, § 6. This is illustrated in exhibit BB of Appendix A.

The multiplicity of each zero is given under the title "MULTIPLICITIES" (exhibit B, Chapter VII).

The initial approximation producing convergence to a root is printed to the right of the corresponding root and headed by "INITIAL APPROXIMATION." The initial approximations may be those supplied by the

user, or generated by the program, or a combination of both (exhibit CC, Appendix A). See Appendix A, § 1 and § 2 for discussion of approximations. The message "SOLVED BY DIRECT METHOD" indicates that the corresponding root or roots was obtained by Subroutine QUAD. See Appendix A, § 5.

If an approximation does not produce convergence within the maximum number of iterations, it is printed under the heading "NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER XXX ITERATIONS." XXX is replaced by the maximum number of iterations. The type of the approximation, that is, initial approximation or altered approximations is given (exhibit AA, Appendix A). See Appendix A, § 1 and § 2 for discussion of approximations.

5. Informative and Error Messages

The output may contain informative or error messages. These are intended as an aid to the user and are described as follows:

"IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(X) = YYY DID NOT CONVERGE. THE PRESENT APPROXIMATION AFTER ZZZ ITERATIONS IS PRINTED BELOW." X is the number of the zero, YYY is the value of the zero before the attempt to improve accuracy, ZZZ is the maximum number of iterations. This message indicates that a zero found before attempting to improve accuracy did not converge sufficiently when being used as an initial approximation on the full (undeflated) polynomial. The current approximation is printed in the list of improved zeros. In many cases, this failure to converge is a result of an ill-conditioned polynomial and this current approximation of the root may be better than its approximation before the attempt to improve accuracy. In most cases, the

polynomial from which this root was first extracted had fewer multiple roots, due to deflations, than the original polynomial.

"THE VALUE OF THE DERIVATIVE AT $XO = XXX$ IS ZERO."

This message is printed as a result of the value of the derivative of the original polynomial at an approximation, XXX , being zero (0). It occurred in the attempt to improve the accuracy of a zero. The previous message is then printed.

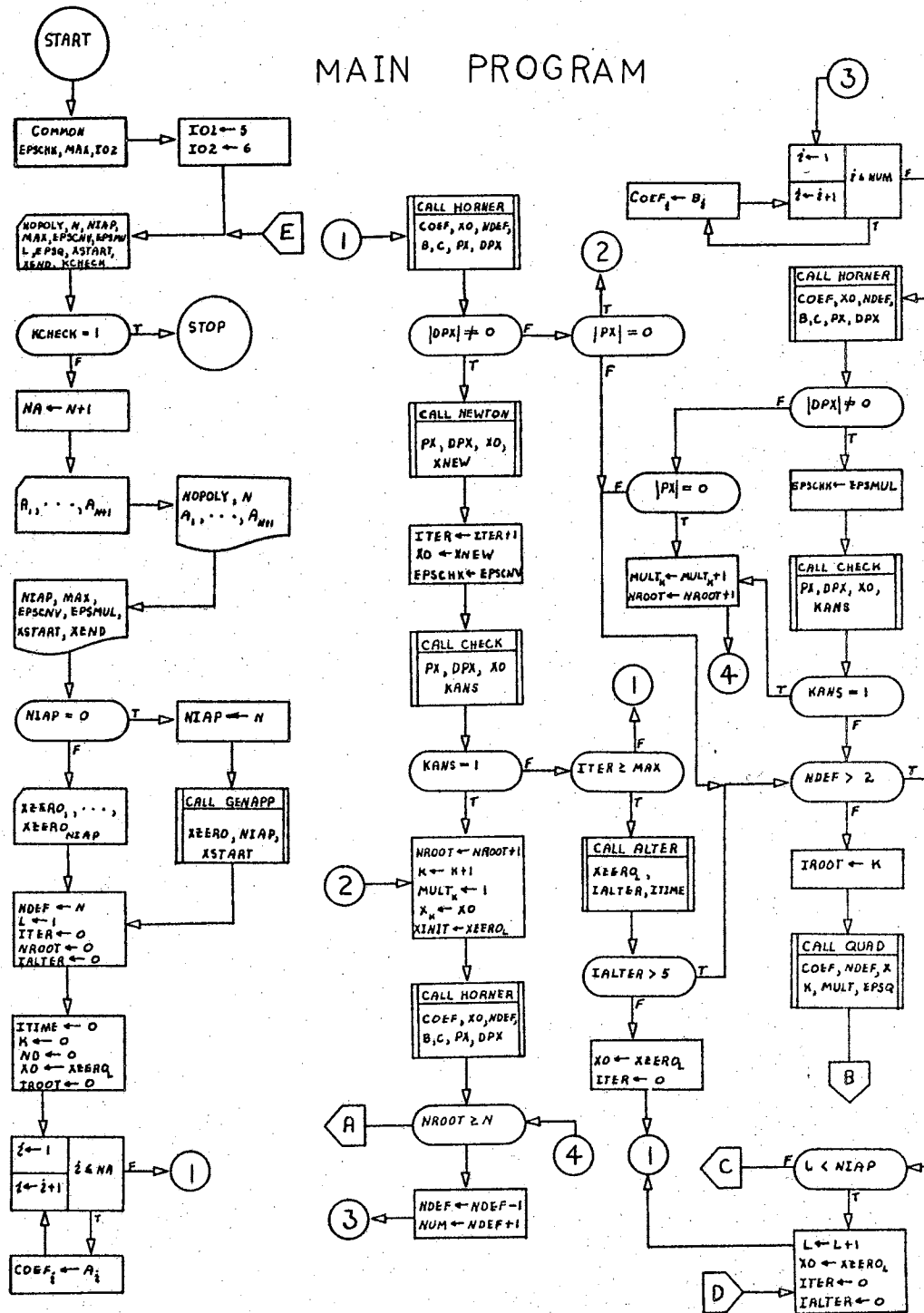


Figure 12. Flow Charts for Newton's Method

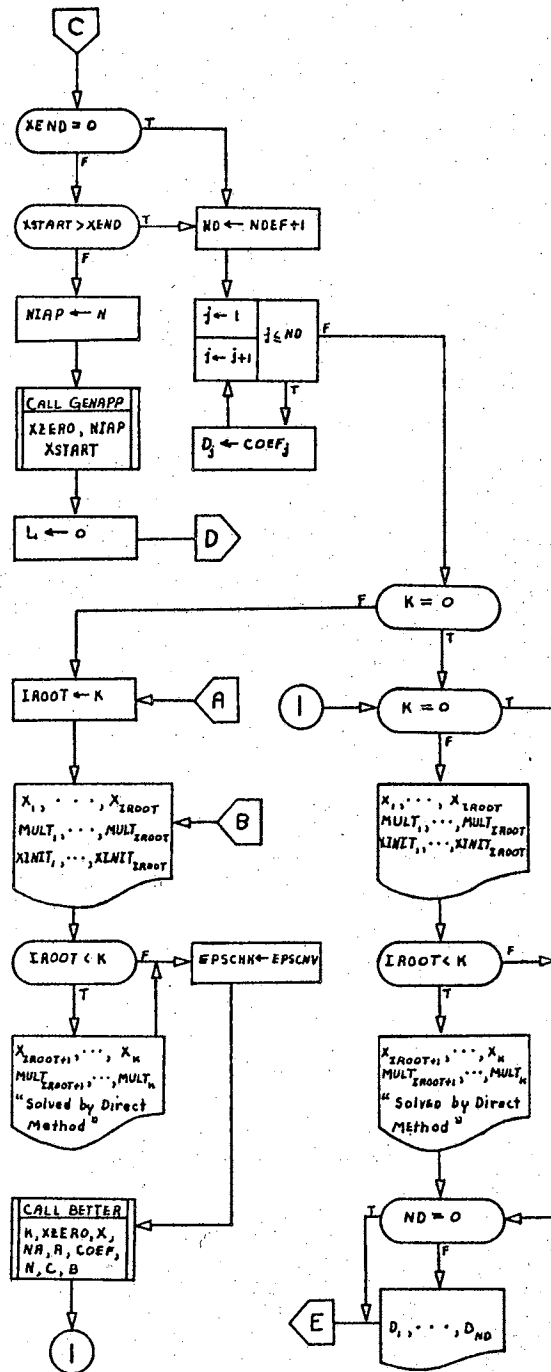


Figure 12. (Continued)

BETTER

CHECK

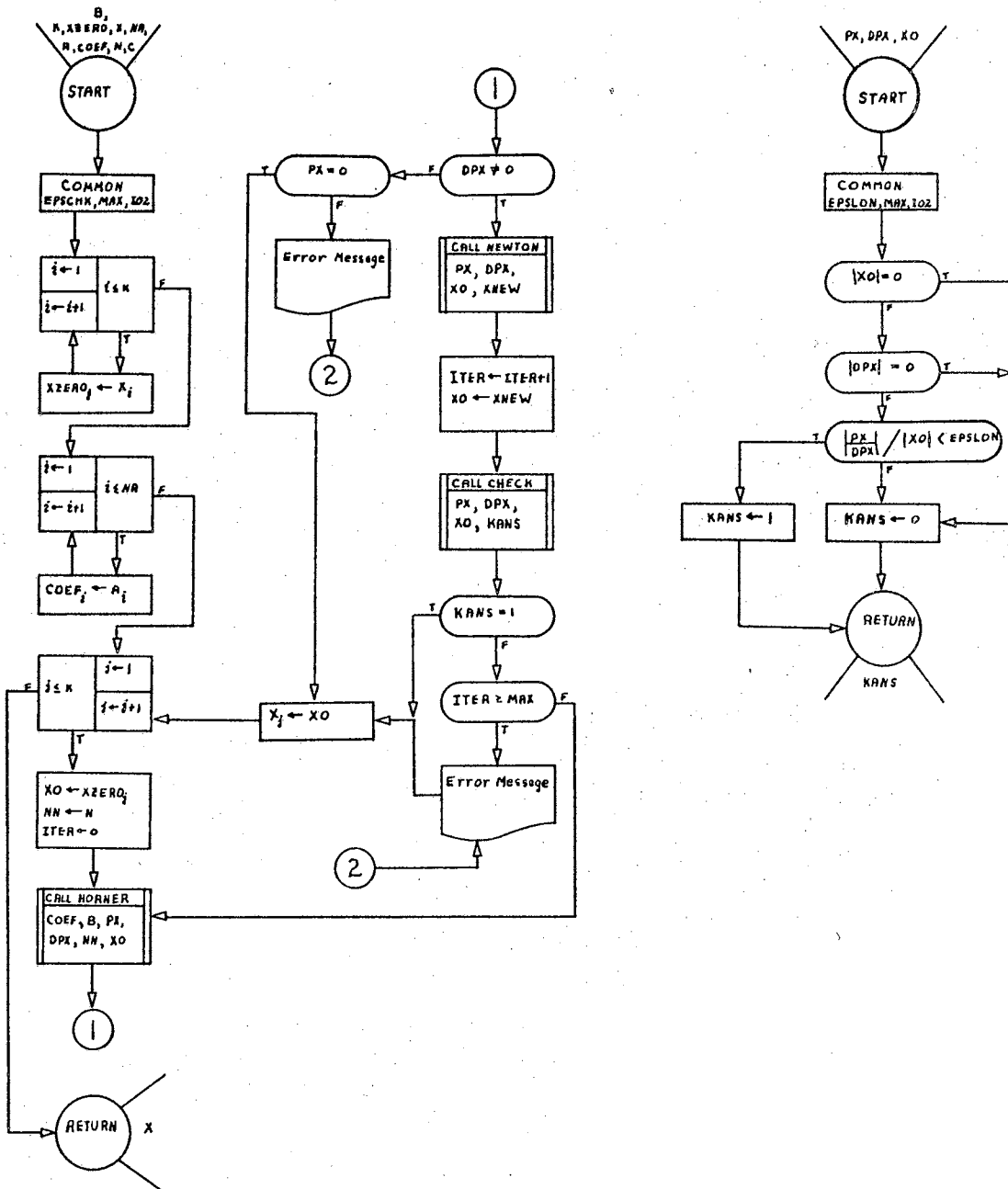
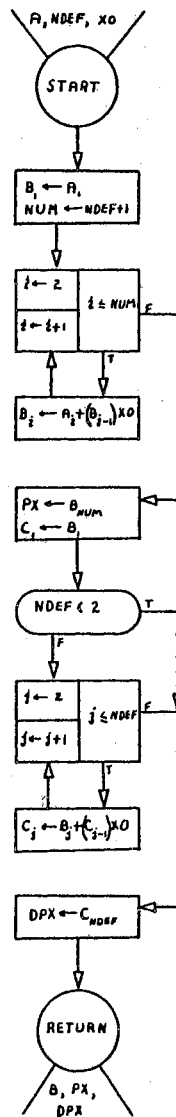


Figure 12. (Continued)

HORNER



NEWTON

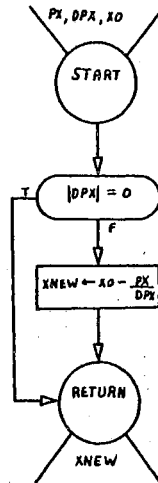
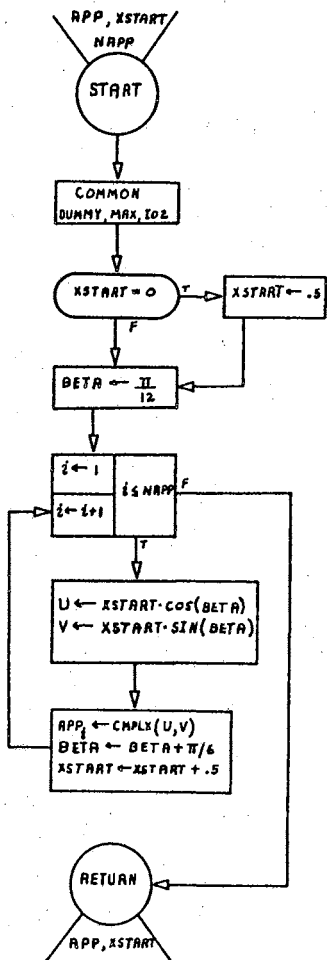


Figure 12. (Continued)

GENAPP



ALTER

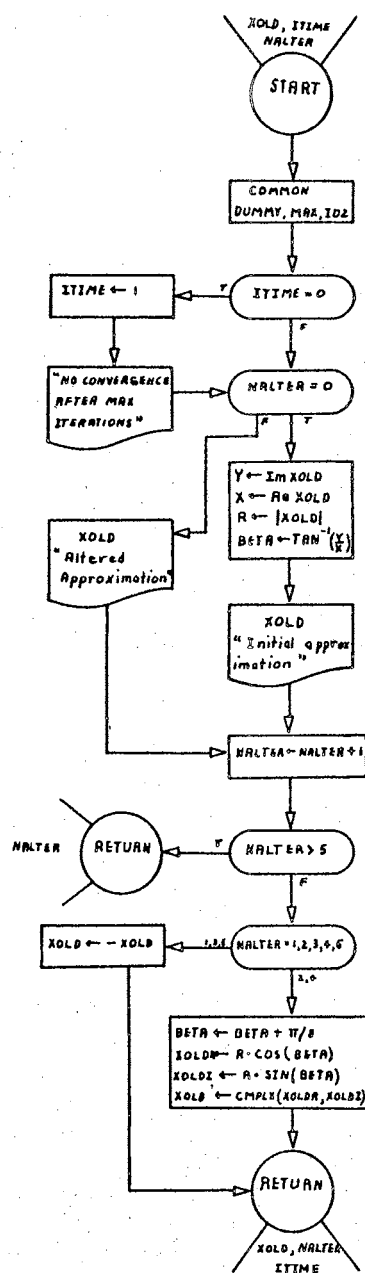
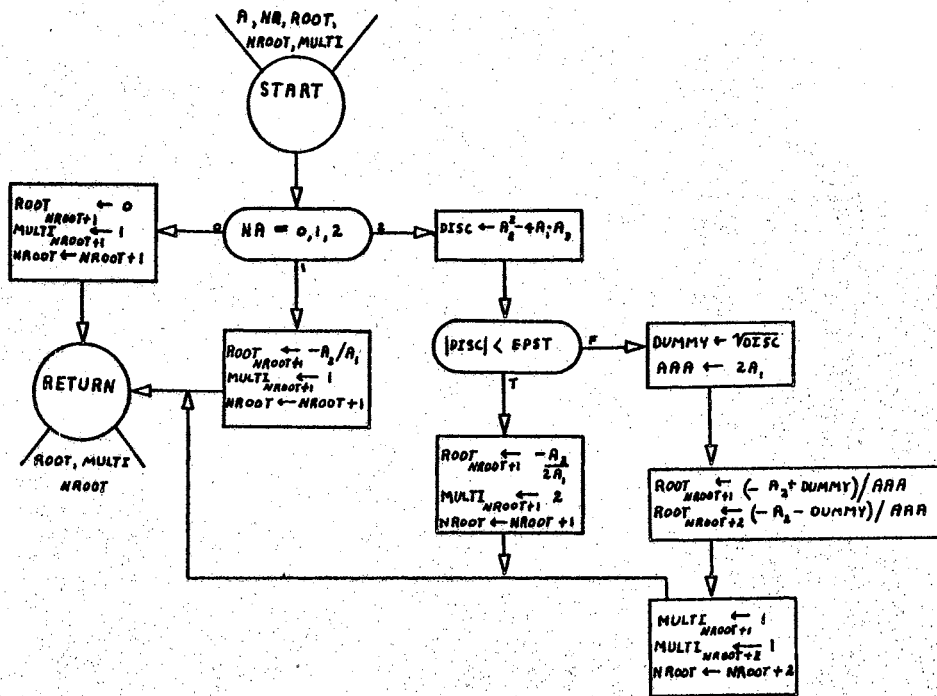


Figure 12. (Continued)

QUAD



COMSQT

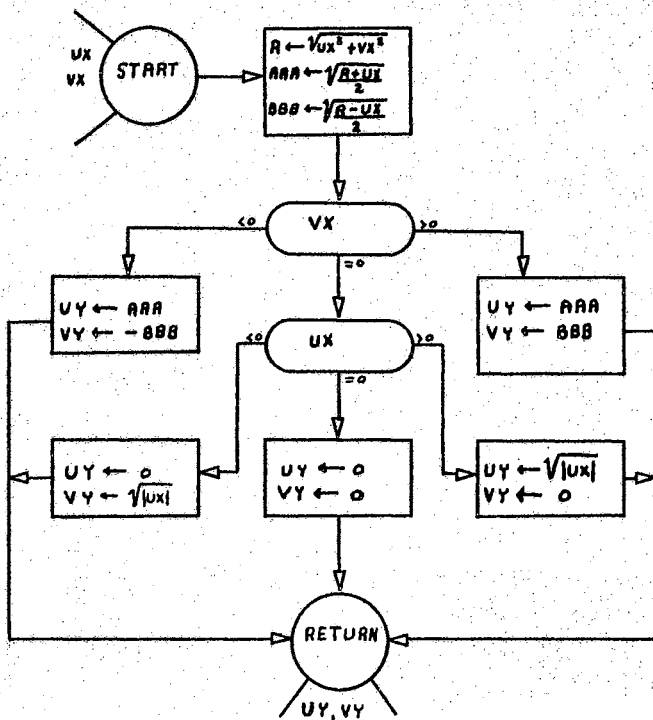


Figure 12. (Continued)

TABLE VIII-A

SINGLE PRECISION PROGRAM FOR NEWTON'S METHOD

```

$JOB 10414
C *****
C *
C * SINGLE PRECISION PROGRAM FOR NEWTON'S METHOD
C *
C *
C * NEWTONS METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
C * POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX-
C * IMATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION
C * FORMULA
C *
C *          X(N+1) = X(N)-P(X(N))/P'(X(N)).
C *
C *****
1  COMPLEX A,XZERO,B,COEF,X,XINIT,C,D,PX,DPX,XNEW,XO
2  DIMENSION A(26),B(26),C(26),D(26),COEF(26),MULT(25),XZERO(25),X(25
3  1),XINIT(25)
4  COMMON EPSCHK,MAX,IO2
5  IO1=5
6  IO2=6
7  1 READ(IO1,1000) NOPOLY,N,NIAP,MAX,EPSCNV,EPSQ,EPSMUL,XSTART,XEND,KC
8  IHECK
9  IF(KCHECK.EQ.1) STOP
10 NA=N+1
11 READ(IO1,1010) (A(I),I=1,NA)
12 WRITE(IO2,1030) NOPOLY,N
13 WRITE(IO2,1040) (I,A(I),I=1,NA)
14 WRITE(IO2,2060)
15 WRITE(IO2,2000) NIAP
16 WRITE(IO2,2010) MAX
17 WRITE(IO2,2020) EPSCNV
18 WRITE(IO2,2030) EPSMUL
19 WRITE(IO2,2040) XSTART
20 WRITE(IO2,2050) XEND
21 IF(NIAP.NE.0) GO TO 3
22 NIAP=N
23 CALL GENAPP(XZERO,NIAP,XSTART)
24 GO TO 4
25 3 READ(IO1,1020) (XZERO(I),I=1,NIAP)
26 4 NDEF=N
27 L=1
28 ITER=0
29 NROOT=0
30 IROOT=0
31 IALTER=0
32 ITIME=0
33 ND=0
34 K=0
35 XO=XZERO(L)
36 DO 5 I=1,NA
37 COEF(I)=A(I)
38 5 CALL HORNER(COEF,B,PX,DPX,NDEF,XO,C)
39 ABPX=CABS(PX)
40 ABDPX=CABS(DPX)
41 IF(ABDPX.NE.0.0) GO TO 20
42 IF(ABPX.EQ.0.0) GO TO 70
43 GO TO 110
44 20 CALL NEWTON(PX,DPX,XO,XNEW)
45 ITER=ITER+1
46 XO=XNEW

```


TABLE VIII-A (Continued)

45	EPSCHK=EPSCNV	
46	CALL CHECK(PX,DPX,XO,KANS)	
47	IF(KANS.EQ.1) GO TO 70	194
48	IF(ITER.GE.MAX) GO TO 40	
49	GO TO 10	208
50	40 CALL ALTER(XZERO(L),IALTER,ITIME)	
51	IF(IALTER.GT.5) GO TO 110	
52	XO=XZERO(L)	
53	ITER=0	244
54	GO TO 10	248
55	60 ND=NDEF+1	
56	DO 65 J=1,ND	
57	65 D(J)=COEF(J)	
58	GO TO 140	
59	70 NROOT=NROOT+1	268
60	K=K+1	272
61	MULT(K)=1	276
62	X(K)=XO	280
63	XINIT(K)=XZERO(L)	288
64	CALL HORNER(COEF,B,PX,DPX,NDEF,XO,C)	
65	80 IF(NROOT.GE.N) GO TO 147	
66	NDEF=NDEF-1	
67	NUM=NDEF+1	
68	DO 105 I=1,NUM	294
69	105 COEF(I)=B(I)	296
70	CALL HORNER(COEF,B,PX,DPX,NDEF,XO,C)	
71	ABPX=CABS(PX)	
72	ABDPX=CABS(DPX)	
73	IF(ABDPX.NE.0.0) GO TO 107	
74	IF(ABPX.EQ.0.0) GO TO 130	
75	GO TO 110	
76	107 CONTINUE	
77	EPSCHK=EPSMUL	
78	CALL CHECK(PX,DPX,XO,KANS)	
79	IF(KANS.EQ.1) GO TO 130	300
80	110 IF(NDEF.GT.2) GO TO 113	
81	IROOT=K	
82	CALL QUAD(COEF,NDEF,X,K,MULT,EPSQ)	
83	GO TO 150	
84	113 IF(L.LT.NIAP) GO TO 115	
85	IF(XEND.EQ.0.0) GO TO 60	
86	IF(XSTART.GT.XEND) GO TO 60	
87	NIAP=N	
88	CALL GENAPP(XZERO,NIAP,XSTART)	
89	L=0	
90	115 L=L+1	
91	XO=XZERO(L)	312
92	ITER=0	316
93	IALTER=0	320
94	GO TO 10	324
95	130 MULT(K)=MULT(K)+1	328
96	NROOT=NROOT+1	332
97	GO TO 80	336
98	140 IF(K.EQ.0) GO TO 160	338
99	147 IROOT=K	
100	150 WRITE(102,1025)	
101	WRITE(102,1050)	380
102	WRITE(102,1060) (I,X(I),MULT(I),XINIT(I),I=1,IROOT)	
103	KKK=IROOT+1	
104	IF(IROOT.LT.K) WRITE(102,1062) (I,X(I),MULT(I),I=KKK,K)	

TABLE VIII-A (Continued)

```

105     EPSCHK=EPSCNV
106     CALL BETTER(K,XZERO,X,NA,A,COEF,N,C,B)
107     160 IF(K.EQ.0) GO TO 170
108         WRITE(IO2,1065)
109         WRITE(IO2,1050)
110         WRITE(IO2,1060) (I,X(I),MULT(I),XINIT(I),I=1,IROOT)
111         KKK=IROOT+1
112         IF(IROOT.LT.K) WRITE(IO2,1062) (I,X(I),MULT(I),I=KKK,K)
113     170 IF(ND.EQ.0) GO TO 1
114         WRITE(IO2,1070)
115         WRITE(IO2,1075) (J,D(J),J=1,ND)
116         GO TO 1
117     1000 FORMAT(3(I2,1X),9X,I3,8X,3(E6.0,1X),13X,2(E7.0,1X),I1)
118     1010 FORMAT(2E30.0)
119     1030 FORMAT(1H1,8X,43HNEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS/9X,18
        1HPOLYNOMIAL NUMBER ,I2,11H OF DEGREE ,I2,///1X,28HTHE COEFFICIENT
        2S OF P(X) ARE/)
120     1040 FORMAT(3X,2HP(,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
121     2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,I2)
122     2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
123     2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,E9.2)
124     2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,E9.2)
125     2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,E9.2)
126     2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,E9.2)
127     2060 FORMAT(///1X)
128     1020 FORMAT(2E30.0)
129     1025 FORMAT(///1X,61HBEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
        1OF P(X) ARE)
130     1050 FORMAT(///1X,13HZEROS OF P(X),38X,14HMULTIPLICITIES,11X,21HINITIAL
        1 APPROXIMATION/)
131     1060 FORMAT(3X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,9X,I2,12X,E14.7
        1,3H + ,E14.7,2H I)
132     1062 FORMAT(3X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,9X,I2,13X,23HSO
        1LVED BY DIRECT METHOD)
133     1065 FORMAT(///1X,61HAFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
        1OF P(X) ARE)
134     1070 FORMAT(///1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO Z
        1EROS WERE FOUND/)
135     1075 FORMAT(3X,2HD(,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
136     END

```

378

396

444

450

TABLE VIII-A (Continued)

```

159      SUBROUTINE CHECK(PX,DPX,XO,KANS)
C      *****
C      *
C      * THIS SUBROUTINE CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
C      * IMATIONS BY TESTING THE EXPRESSION
C      * ABSOLUTE VALUE OF (P(X(N))/P'(X(N)))/ABSOLUTE VALUE OF X(N+1).
C      * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
C      *
C      *****
160      COMPLEX PX,DPX,XO
161      COMMON EPSLON,MAX,I02
162      IF(CABS(XO).EQ.0.) GO TO 25
163      DDD=CABS(DPX)
164      IF(DDD.EQ.0.0) GO TO 25
165      IF(CABS(PX/DPX)/CABS(XO).LT.EPSLON) GO TO 10
166      KANS=0
167      RETURN
168      10 KANS=1
169      RETURN
170      25 KANS=0
171      RETURN
172      END

```

748
760
764
768
772
780

TABLE VIII-A (Continued)

```

173      SUBROUTINE BETTER(K,XZERO,X,NA,A,COEF,N,C,B)
C      *****
C      *
C      * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND
C      * BY USING THEM AS INITIAL APPROXIMATIONS WITH NEWTON'S METHOD APPLIED TO
C      * THE FULL, UNDEFLATED POLYNOMIAL.
C      *
C      *****
174      COMPLEX XZERO,X,A,COEF,C,B,XD,PX,DPX,XNEW      804
175      DIMENSION XZERO(25),X(25),A(26),COEF(26),C(26),B(26)
176      COMMON EPSCHK,MAX,I02
177      DO 10 I=1,K      812
178      10 XZERO(I)=X(I)      815
179      DO 20 I=1,NA      820
180      20 COEF(I)=A(I)      824
181      DO 50 J=1,K      828
182      XD=XZERO(J)      832
183      NN=N      834
184      ITER=0      836
185      30 CALL HORNER(COEF,B,PX,DPX,NN,XD,C)
186      ABPX=CABS(PX)
187      ABDPX=CABS(DPX)
188      IF(ABDPX.NE.0.0) GO TO 33
189      IF(ABPX.EQ.0.0) GO TO 40
190      GO TO 34
191      33 CALL NEWTON(PX,DPX,XD,XNEW)
192      ITER=ITER+1      856
193      XO=XNEW      860
194      CALL CHECK(PX,DPX,XD,KANS)
195      IF(KANS.EQ.1) GO TO 40      844
196      IF(ITER.GE.MAX) GO TO 35
197      GO TO 30      864
198      34 WRITE(I02,1112) XD
199      35 WRITE(I02,100) J,XZERO(J)
200      WRITE(I02,200) MAX
201      40 X(IJ)=XD      868
202      50 CONTINUE      872
203      RETURN      876
204      1112 FORMAT(1H0,36H THE VALUE OF THE DERIVATIVE AT XD = ,E14.7,3H + ,E14
1.7,2H I,10H IS ZERO.)
205      100 FORMAT(42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,E14
1.7,3H + ,E14.7,2H I,18H DID NOT CONVERGE.)
206      200 FORMAT(33H THE PRESENT APPROXIMATION AFTER ,I3,29H ITERATIONS IS P
RINTED BELOW.)
207      END      880

```

TABLE VIII-A (Continued)

```

208      SUBROUTINE GENAPP(APP,NAPP,XSTART)
C      *****
C      *
C      * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE *
C      * DEGREE OF THE ORIGINAL POLYNOMIAL. *
C      *
C      *****
209      COMPLEX APP
210      COMPLEX CMLX
211      DIMENSION APP(25)
212      COMMON DUMMY,MAX,IO2
213      IF(XSTART.EQ.0.0) XSTART=0.5
214      BETA=0.2617994
215      DO 10 I=1,NAPP
216      U=XSTART*COS(BETA)
217      V=XSTART*SIN(BETA)
218      APP(I)=CMLX(U,V)
219      BETA=BETA+0.5235988
220 10 XSTART=XSTART+0.5
221      RETURN
222      END

223      SUBROUTINE ALTER(XOLD,NALTER,ITIME)
C      *****
C      *
C      * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO *
C      * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
C      *
C      *****
224      COMPLEX XOLD
225      COMPLEX CMLX
226      COMMON DUMMY,MAX,IO2
227      IF(ITIME.NE.0) GO TO 5
228      ITIME=1
229      WRITE(IO2,1010) MAX
230 5 IF(NALTER.EQ.0) GO TO 10
231      WRITE(IO2,1000) XOLD
232      GO TO 20
233 10 Y=AIMAG(XOLD)
234      X=REAL(XOLD)
235      R=CABS(XOLD)
236      BETA=ATAN2(Y,X)
237      WRITE(IO2,1020) XOLD
238 20 NALTER=NALTER+1
239      IF(NALTER.GT.5) RETURN
240      GO TO (30,40,30,40,30),NALTER
241 30 XOLD=-XOLD
242      GO TO 50
243 40 BETA=BETA+1.0471976
244      XOLDR=R*COS(BETA)
245      XOLDI=R*SIN(BETA)
246      XOLD=CMLX(XOLDR,XOLDI)
247 50 RETURN
248 1000 FORMAT(1X,E14.7,3H + ,E14.7,2H I,10X,21HALTERED APPROXIMATION)
249 1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
250 1020 FORMAT(//1X,E14.7,3H + ,E14.7,2H I,10X,21HINITIAL APPROXIMATION)
251      END

```

TABLE VIII-A (Continued)

```

252      SUBROUTINE QUAD(A,NA,ROOT,NROOT,MULTI,EPST)
C      *****
C      *
C      * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES
C      * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE
C      * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.
C      *
C      *****
253      COMPLEX A,DISC,ROOT,DUMMY,AAA
254      COMPLEX CSQRT
255      DIMENSION A(26),ROOT(25),MULTI(25)
256      IF(NA.EQ.2) GO TO 7
257      IF(NA.EQ.1) GO TO 5
258      ROOT(NROOT+1)=0.0
259      MULTI(NROOT+1)=1
260      NROOT=NROOT+1
261      GO TO 50
262      5 ROOT(NROOT+1)=-A(2)/A(1)
263      MULTI(NROOT+1)=1
264      NROOT=NROOT+1
265      GO TO 50
266      7 DISC=A(2)*A(2)-(4.0*A(1)*A(3))
267      BBB=CABS(DISC)
268      IF(BBB.LT.EPST) GO TO 10
269      DUMMY=CSQRT(DISC)
270      AAA=2.0*A(1)
271      ROOT(NROOT+1)=(-A(2)+DUMMY)/AAA
272      ROOT(NROOT+2)=(-A(2)-DUMMY)/AAA
273      MULTI(NROOT+1)=1
274      MULTI(NROOT+2)=1
275      NROOT=NROOT+2
276      GO TO 50
277      10 ROOT(NROOT+1)=(-A(2))/(2.0*A(1))
278      MULTI(NROOT+1)=2
279      NROOT=NROOT+1
280      50 RETURN
281      END

```

\$ENTRY

TABLE VIII-B

DOUBLE PRECISION PROGRAM FOR NEWTON'S METHOD

```

$JOB 10414
C *****
C *
C * DOUBLE PRECISION PROGRAM FOR NEWTON'S METHOD
C *
C *
C * NEWTONS METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
C * POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX-
C * IMATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION
C * FORMULA
C *
C *  $x(N+1) = x(N) - P(x(N))/P'(x(N))$ .
C *
C *****
1  DOUBLE PRECISION RA,VA,RXZERO,VXZERO,VB,VB,RCOEF,VCOEF,RX,VX,RXINI
   1T,VXINIT,RC,VC,RO,VD,RPX,VPX,RDPX,VDPX,RXNEW,VXNEW,RXO,VXO,EPSCCHK,
   2EPSCNV,EPSQ,EPSMUL,XSTART,XEND,ABPX,ABDPX
2  DOUBLE PRECISION DSQRT
3  DIMENSION RA(26),VA(26),RB(26),VB(26),RC(26),VC(26),RD(26),VD(26),
   1RCOEF(26),VCOEF(26),MULT(25),RXZERO(25),VXZERO(25),RX(25),VX(25),R
   2XINIT(25),VXINIT(25)
4  COMMON EPSCCHK,MAX,IO2
5  IO1=5
6  IO2=6
7  1 READ(IO1,1000) NOPOLY,N,NIAP,MAX,EPSCNV,EPSQ,EPSMUL,XSTART,XEND,KC
   1HECK
8  IF(KCHECK.EQ.1) STOP
9  NA=N+1
10 READ(IO1,1010) (RA(I),VA(I),I=1,NA)
11 WRITE(IO2,1030) NOPOLY,N
12 WRITE(IO2,1040) (I,RA(I),VA(I),I=1,NA)
13 WRITE(IO2,2060)
14 WRITE(IO2,2000) NIAP
15 WRITE(IO2,2010) MAX
16 WRITE(IO2,2020) EPSCNV
17 WRITE(IO2,2030) EPSMUL
18 WRITE(IO2,2040) XSTART
19 WRITE(IO2,2050) XEND
20 IF(NIAP.NE.0) GO TO 3
21 NIAP=N
22 CALL GENAPP(RXZERO,VXZERO,NIAP,XSTART)
23 GO TO 4
24 3 READ(IO1,1020) (RXZERO(I),VXZERO(I),I=1,NIAP)
25 4 NDEF=N
26 L=1
27 ITER=0
28 NROOT=0
29 IROOT=0
30 ITIME=0
31 ND=0
32 IALTER=0
33 K=0
34 RXO=RXZERO(L)
35 VXO=VXZERO(L)
36 DO 5 I=1,NA
37 RCOEF(I)=RA(I)
38 5 VCOEF(I)=VA(I)
39 10 CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,VB,VB,RC,VC,RPX,VPX,RDPX,VDPX
   1)
40 ABPX=DSQRT(RPX*RPX+VPX*VPX)

```


TABLE VIII-B (Continued)

41	ABDPX=DSQRT(RDPX*SDPX+VDPX*VDPX)	
42	IF(ABDPX.NE.0.0) GO TO 20	
43	IF(ABPX.EQ.0.0) GO TO 70	
44	GO TO 110	
45	20 CALL NEWTON(RPX,VPX,SDPX,VDPX,RXO,VXO,RXNEW,VXNEW)	
46	ITER=ITER+1	200
47	RXO=RXNEW	204
48	VXO=VXNEW	205
49	EPSCHK=EPSCNV	
50	CALL CHECK(RPX,VPX,SDPX,VDPX,RXO,VXO,KANS)	
51	IF(KANS.EQ.1) GO TO 70	194
52	IF(ITER.GE.MAX) GO TO 40	
53	GO TO 10	208
54	40 CALL ALTER(RXZERO(L),VXZERO(L),IALTER,ITIME)	
55	IF(IALTER.GT.5) GO TO 110	
56	RXO=RXZERO(L)	
57	VXO=VXZERO(L)	
58	ITER=0	244
59	GO TO 10	248
60	60 NDEF=NDEF+1	
61	DO 65 J=1,NDEF	
62	RD(J)=RCOEF(J)	
63	65 VD(J)=VCOEF(J)	
64	GO TO 140	
65	70 NROOT=NROOT+1	268
66	K=K+1	272
67	MULT(K)=1	276
68	RX(K)=RXO	280
69	VX(K)=VXO	281
70	RXINIT(K)=RXZERO(L)	288
71	VXINIT(K)=VXZERO(L)	289
72	CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,VB,RC,VC,RPX,VPX,SDPX,VDPX 1)	
73	80 IF(NROOT.GE.N) GO TO 147	
74	NDEF=NDEF-1	
75	NUM=NDEF+1	
76	DO 105 I=1,NUM	294
77	RCOEF(I)=RB(I)	296
78	105 VCOEF(I)=VB(I)	297
79	CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,VB,RC,VC,RPX,VPX,SDPX,VDPX 1)	
80	ABPX=DSQRT(RPX*SDPX+VPX*VPX)	
81	ABDPX=DSQRT(RDPX*SDPX+VDPX*VDPX)	
82	IF(ABDPX.NE.0.0) GO TO 107	
83	IF(ABPX.EQ.0.0) GO TO 130	
84	GO TO 110	
85	107 CONTINUE	
86	EPSCHK=EPSMUL	
87	CALL CHECK(RPX,VPX,SDPX,VDPX,RXO,VXO,KANS)	
88	IF(KANS.EQ.1) GO TO 130	300
89	110 IF(NDEF.GT.2) GO TO 113	
90	IROOT=K	
91	CALL QUAD(RCOEF,VCOEF,NDEF,RX,VX,K,MULT,EPSQ)	
92	GO TO 150	
93	113 IF(L.LT.NIAP) GO TO 115	
94	IF(XEND.EQ.0.0) GO TO 60	
95	IF(XSTART.GT.XEND) GO TO 60	
96	NIAP=N	
97	CALL GENAPP(RXZERO,VXZERO,NIAP,XSTART)	
98	L=0	

TABLE VIII-B (Continued)

99	115 L=L+1	
100	RXD=RXZERO(L)	312
101	VXD=VXZERO(L)	313
102	ITER=0	316
103	IALTER=0	320
104	GO TO 10	324
105	130 MULT(K)=MULT(K)+1	328
106	NROOT=NROOT+1	332
107	GO TO 80	336
108	140 IF(K.EQ.0) GO TO 160	338
109	147 IROOT=K	
110	150 WRITE(IO2,1025)	
111	WRITE(IO2,1050)	
112	WRITE(IO2,1060) (I,RX(I),VX(I),MULT(I),RXINIT(I),VXINIT(I),I=1,IRO 10T)	
113	KKK=IROOT+1	
114	IF(IROOT.LT.K) WRITE(IO2,1062) (I,RX(I),VX(I),MULT(I),I=KKK,K)	
115	EPSCNK=EPSCNV	
116	CALL BETTER(K,RXZERO,VXZERO,RX,VX,NA,RA,VA,RCOEF,VCOEF,N,RC,VC,VB, 1VB)	
117	160 IF(K.EQ.0) GO TO 170	
118	WRITE(IO2,1065)	
119	WRITE(IO2,1050)	
120	WRITE(IO2,1060) (I,RX(I),VX(I),MULT(I),RXINIT(I),VXINIT(I),I=1,IRO 10T)	
121	KKK=IROOT+1	
122	IF(IROOT.LT.K) WRITE(IO2,1062) (I,RX(I),VX(I),MULT(I),I=KKK,K)	
123	170 IF(ND.EQ.0) GO TO 1	
124	WRITE(IO2,1070)	
125	WRITE(IO2,1075) (J,RD(J),VD(J),J=1,ND)	396
126	GO TO 1	444
127	1000 FORMAT(3(I2,1X),9X,I3,8X,3(D6.0,1X),13X,2(D7.0,1X),I1)	
128	1010 FORMAT(2D30.0)	
129	1030 FORMAT(1H1,8X,43HNEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS/9X,18 1HPOLYNOMIAL NUMBER ,I2,11H OF DEGREE ,I2,///1X,28H THE COEFFICIENT 2S OF P(X) ARE/)	
130	1040 FORMAT(3X,2HP(,I2,4H) = ,D23.16,3H + ,D23.16,2H I)	
131	1020 FORMAT(2D30.0)	
132	1025 FORMAT(///1X,61HBEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS 1OF P(X) ARE)	
133	1050 FORMAT(///2X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL 1 APPROXIMATION//)	
134	1060 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,7X,D23. 116,3H + ,D23.16,2H I)	
135	1062 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,8X,23HS 1OLVED BY DIRECT METHOD)	
136	1065 FORMAT(///1X,61HAFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS 1OF P(X) ARE)	
137	1070 FORMAT(///1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO Z 1EROS WERE FOUND/)	
138	1075 FORMAT(3X,2HD(,I2,4H) = ,D23.16,3H + ,D23.16,2H I)	
139	2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,I2)	
140	2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)	
141	2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)	
142	2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)	
143	2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)	
144	2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)	
145	2060 FORMAT(///1X)	
146	END	450

TABLE VIII-B (Continued)

```

147      SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
C      *****
C      *
C      * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE *
C      * DEGREE OF THE ORIGINAL POLYNOMIAL. *
C      *
C      *****
148      DOUBLE PRECISION APPR,APPI,XSTART,DUMMY,BETA
149      DOUBLE PRECISION DCOS,DSIN
150      DIMENSION APPR(25),APPI(25)
151      COMMON DUMMY,MAX,IO2
152      IF(XSTART.EQ.0.0) XSTART=0.5
153      BETA=0.2617994
154      DO 10 I=1,NAPP
155      APPR(I)=XSTART*DCOS(BETA)
156      APPI(I)=XSTART*DSIN(BETA)
157      BETA=BETA+0.5235988
158 10 XSTART=XSTART+0.5
159      RETURN
160      END

161      SUBROUTINE ALTER(XOLDR,XOLDI,NALTER,ITIME)
C      *****
C      *
C      * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO *
C      * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
C      *
C      *****
162      DOUBLE PRECISION XOLDR,XOLDI,DUMMY,ABXOLD,BETA
163      DOUBLE PRECISION DCOS,DSIN,DATAN2
164      DOUBLE PRECISION DSQRT
165      COMMON DUMMY,MAX,IO2
166      IF(ITIME.NE.0) GO TO 5
167      ITIME =1
168      WRITE(IO2,1010) MAX
169      5 IF(NALTER.EQ.0) GO TO 10
170      WRITE(IO2,1000) XOLDR,XOLDI
171      GO TO 20
172 10 ABXOLD=DSQRT(XOLDR*XOLDR+XOLDI*XOLDI)
173      BETA=DATAN2(XOLDI,XOLDR)
174      WRITE(IO2,1020) XOLDR,XOLDI
175 20 NALTER=NALTER+1
176      IF(NALTER.GT.5) RETURN
177      GO TO (30,40,30,40,30),NALTER
178 30 XOLDR=-XOLDR
179      XOLDI=-XOLDI
180      GO TO 50
181 40 BETA=BETA+1.0471976
182      XOLDR=ABXOLD*DCOS(BETA)
183      XOLDI=ABXOLD*DSIN(BETA)
184 50 RETURN
185 1000 FORMAT(1X,D23.16,3H + ,D23.16,2H I,10X,21HALTERED APPROXIMATION)
186 1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
ITER ,I3,12H ITERATIONS.//)
187 1020 FORMAT(1X,D23.16,3H + ,D23.16,2H I,10X,21HINITIAL APPROXIMATION)
188      END

```

TABLE VIII-B (Continued)

```

189      SUBROUTINE QUAD(UA,VA,NA,UROOT,VROOT,NROOT,MULTI,EPST)
C      *****
C      *
C      * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
C      * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE *
C      * QUADRATIC IS DONE USING THE QUADRATIC FORMULA. *
C      *
C      *****
190      DOUBLE PRECISION UA,VA,UROOT,VROOT,BBB,UAAA,VAAA,UDISC,VDISC,UDUMM
        LY,VDUMMY,RDUMMY,SDUMMY,EPST,UBBB,VBBB
191      DOUBLE PRECISION DSQRT
192      DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),MULTI(25)
193      IF(NA.EQ.2) GO TO 7
194      IF(NA.EQ.1) GO TO 5
195      UROOT(NROOT+1)=0.0
196      VROOT(NROOT+1)=0.0
197      MULTI(NROOT+1)=1
198      NROOT=NROOT+1
199      GO TO 50
200      5 BBB=UA(1)*UA(1)+VA(1)*VA(1)
        UROOT(NROOT+1)=(-UA(2)*UA(1)-VA(2)*VA(1))/BBB
202      VROOT(NROOT+1)=(-VA(2)*UA(1)+UA(2)*VA(1))/BBB
203      MULTI(NROOT+1)=1
204      NROOT=NROOT+1
205      GO TO 50
206      7 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-{4.0*(UA(1)*UA(3)-VA(1)*VA(3))}
        VDISC=(VA(2)*UA(2)+UA(2)*VA(2))-{4.0*(VA(1)*UA(3)+UA(1)*VA(3))}
208      BBB=DSQRT(UDISC*UDISC+VDISC*VDISC)
209      IF(BBB.LT.EPST) GO TO 10
210      CALL COMSQT(UDISC,VDISC,UDUMMY,VDUMMY)
211      UBBB=-UA(2)+UDUMMY
212      VBBB=-VA(2)+VDUMMY
213      RDUMMY=-UA(2)-UDUMMY
214      SDUMMY=-VA(2)-VDUMMY
215      UAAA=2.0*UA(1)
216      VAAA=2.0*VA(1)
217      BBB=UAAA*UAAA+VAAA*VAAA
218      UROOT(NROOT+1)=(UBBB*UAAA+VBBB*VAAA)/BBB
219      VROOT(NROOT+1)=(VBBB*UAAA-UBBB*VAAA)/BBB
220      UROOT(NROOT+2)=(RDUMMY*UAAA+SDUMMY*VAAA)/BBB
221      VROOT(NROOT+2)=(SDUMMY*UAAA-RDUMMY*VAAA)/BBB
222      MULTI(NROOT+1)=1
223      MULTI(NROOT+2)=1
224      NROOT=NROOT+2
225      GO TO 50
226      10 UAAA=2.0*UA(1)
        VAAA=2.0*VA(1)
228      BBB=UAAA*UAAA+VAAA*VAAA
229      UROOT(NROOT+1)=(-UA(2)*UAAA-VA(2)*VAAA)/BBB
230      VROOT(NROOT+1)=(-VA(2)*UAAA+UA(2)*VAAA)/BBB
231      MULTI(NROOT+1)=2
232      NROOT=NROOT+1
233      50 RETURN
234      END

```

TABLE VIII-B (Continued)

```

235      SUBROUTINE COMSQT(UX,VX,UY,VY)
C      *****
C      *
C      * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
C      *
C      *****
236      DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
237      DOUBLE PRECISION DSQRT,DABS
238      R=DSQRT(UX*UX+VX*VX)
239      AAA=DSQRT(DABS((R+UX)/2.0))
240      BBB=DSQRT(DABS((R-UX)/2.0))
241      IF(VX) 10,20,30
242      10 UY=AAA
243         VY=-1.0*BBB
244         GO TO 100
245      20 IF(UX) 40,50,60
246      30 UY=AAA
247         VY=BBB
248         GO TO 100
249      40 DUMMY=DABS(UX)
250         UY=0.0
251         VY=DSQRT(DUMMY)
252         GO TO 100
253      50 UY=0.0
254         VY=0.0
255         GO TO 100
256      60 DUMMY=DABS(UX)
257         UY=DSQRT(DUMMY)
258         VY=0.0
259      100 RETURN
260      END

261      SUBROUTINE HORNER(RA,VA,RX0,VX0,NDEF,VB,VB,RC,VC,RPX,VPX,RDPX,VDPX)
C      *****
C      *
C      * HORNER'S METHOD COMPUTES THE VALUE OF A POLYNOMIAL P(X) AT A POINT D AND *
C      * ITS DERIVATIVE AT D. SYNTHETIC DIVISION IS USED TO DEFLATE THE *
C      * POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D).
C      *
C      *****
1)
262      DOUBLE PRECISION VDPX,RX0,VX0,VB,VB,RC,VC,RPX,VPX,RDPX,RA,VA
263      DIMENSION RA(26),VA(26),RB(26),VB(26),RC(26),VC(26)
264      RB(1)=RA(1)
265      VB(1)=VA(1)
266      NUM=NDEF+1
267      DO 10 I=2,NUM
268         RB(I)=RA(I)+(RB(I-1)*RX0-VB(I-1)*VX0)
269      10 VB(I)=VA(I)+(VB(I-1)*RX0+RB(I-1)*VX0)
270      RPX=RB(NUM)
271      VPX=VB(NUM)
272      RC(1)=RB(1)
273      VC(1)=VB(1)
274      IF(NDEF.LT.2) GO TO 25
275      DO 20 J=2,NDEF
276         RC(J)=RB(J)+(RC(J-1)*RX0-VC(J-1)*VX0)
277      20 VC(J)=VB(J)+(VC(J-1)*RX0+RC(J-1)*VX0)
278      25 RDPX=RC(NDEF)
279         VDPX=VC(NDEF)
280      RETURN
281      END

```

TABLE VIII-B (Continued)

```

282      SUBROUTINE NEWTON(RPX,VPX, RDPX, VDPX, RXO, VXO, RXNEW, VXNEW)      600
C      *****
C      *
C      * THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-
C      * IMATION BY USING THE ITERATION FORMULA
C      *
C      *       $X\{N+1\} = X\{N\} - P\{X\{N\}\} / P'\{X\{N\}\}.$ 
C      *
C      *****
283      DOUBLE PRECISION RPX,VPX, RDPX, VDPX, RXO, VXO, RXNEW, VXNEW, ARG
284      DOUBLE PRECISION DSQRT
285      DOUBLE PRECISION DDD
286      ARG= RDPX* RDPX + VDPX* VDPX
287      DDD= DSQRT( ARG )
288      IF( DDD.EQ.0.0 ) RETURN
289      RXNEW= RXO - { ( RPX* RDPX + VPX* VDPX ) / ARG }
290      VXNEW= VXO - { ( VPX* RDPX - RPX* VDPX ) / ARG }
291      RETURN
292      END
293
293      SUBROUTINE CHECK( RPX, VPX, RDPX, VDPX, RXO, VXO, KANS )
C      *****
C      *
C      * THIS SUBROUTINE CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
C      * IMATIONS BY TESTING THE EXPRESSION
C      * ABSOLUTE VALUE OF { P{X{N}} / P'{X{N}} } / ABSOLUTE VALUE OF X{N+1}.
C      * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
C      *
C      *****
294      DOUBLE PRECISION RPX,VPX, RDPX, VDPX, RXO, VXO, ABSXO, ABSQUO, RDUMMY, VDU
C      LMZY, EPS
295      DOUBLE PRECISION ARG
296      DOUBLE PRECISION DSQRT
297      DOUBLE PRECISION DDD
298      COMMON EPS, MAX, IOZ
299      ABSXO= DSQRT( RXO* RXO + VXO* VXO )
300      IF( ABSXO.EQ.0. ) GO TO 25
301      ARG= RDPX* RDPX + VDPX* VDPX
302      DDD= DSQRT( ARG )
303      IF( DDD.EQ.0.0 ) GO TO 25
304      RDUMMY= ( RPX* RDPX + VPX* VDPX ) / ARG
305      VDUMMY= ( VPX* RDPX - RPX* VDPX ) / ARG
306      ABSQUO= DSQRT( RDUMMY* RDUMMY + VDUMMY* VDUMMY )
307      IF( ABSQUO/ABSXO.LT.EPS ) GO TO 10
308      KANS=0
309      RETURN
310      10 KANS=1
311      RETURN
312      25 KANS=0
313      RETURN
314      END

```

TABLE VIII-B (Continued)

```

315      SUBROUTINE BETTER(K,RXZERO,VXZERO,RX,VX,NA,RA,VA,RCOEF,VCDEF,N,RC,
          IVC,VB,VB)
C      *****
C      *
C      * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND *
C      * BY USING THEM AS INITIAL APPROXIMATIONS WITH NEWTON'S METHOD APPLIED TO *
C      * THE FULL, UNDEFLATED POLYNOMIAL. *
C      *
C      *****
316      DOUBLE PRECISION RXZERO,VXZERO,RX,VX,RA,VA,RCOEF,VCDEF,RC,VC,VB,VB      805
          I,RXO,VXO,RPX,VPX, RDPX, VDPX, RXNEW, VXNEW, EPS
317      DOUBLE PRECISION DSQRT
318      DIMENSION RXZERO(25),VXZERO(25),RX(25),VX(25),RA(26),VA(26),RCOEF(
          I26),VCDEF(26),RC(26),VC(26),RB(26),VB(26)
319      DOUBLE PRECISION ABPX,ABDPX
320      COMMON EPS,MAX,IO2
321      DO 10 I=1,K                                812
322      RXZERO(I)=RX(I)                            815
323      10 VXZERO(I)=VX(I)                          816
324      DO 20 I=1,NA
325      RCOEF(I)=RA(I)                                824
326      20 VCDEF(I)=VA(I)                            825
327      DO 50 J=1,K                                828
328      RXO=RXZERO(J)                                832
329      VXO=VXZERO(J)                                833
330      NN=N                                          834
331      ITER=0                                        836
332      30 CALL HORNER(RCOEF,VCDEF,RXO,VXO,NN,RB,VB,RC,VC,RPX,VPX, RDPX, VDPX)
333      ABPX=DSQRT(RPX*RPX+VPX*VPX)
334      ABDPX=DSQRT(RDPX* RDPX+VDPX*VDPX)
335      IF(ABDPX.NE.0.0) GO TO 33
336      IF(ABPX.EQ.0.0) GO TO 40
337      GO TO 34
338      33 CALL NEWTON(RPX,VPX, RDPX, VDPX, RXO, VXO, RXNEW, VXNEW)
339      ITER=ITER+1                                856
340      RXO=RXNEW                                  860
341      VXO=VXNEW                                  861
342      CALL CHECK(RPX,VPX, RDPX, VDPX, RXO, VXO, KANS)
343      IF(KANS.EQ.1) GO TO 40                      844
344      IF(ITER.GE.MAX) GO TO 35
345      GO TO 30                                    864
346      34 WRITE(IO2,1112) RXO,VXO
347      35 WRITE(IO2,100) J,RXZERO(J),VXZERO(J)
348      WRITE(IO2,200) MAX
349      40 RX(J)=RXO                                870
350      VX(J)=VXO                                  871
351      50 CONTINUE                                872
352      RETURN                                      876
353      1112 FORMAT(1H0,36H THE VALUE OF THE DERIVATIVE AT X0 = ,D23.16,3H + ,D2
          13.16,2H 1,10H IS ZERO.)
354      100 FORMAT(42H0 IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,D23
          1.16,3H + ,D23.16,2H 1,18H DID NOT CONVERGE.)
355      200 FORMAT(33H THE PRESENT APPROXIMATION AFTER ,I3,29H ITERATIONS IS P
          RINTED BELOW.)
356      END                                          880

```

\$ENTRY

APPENDIX C

MULLER'S METHOD

1. Use of the Programs

Two programs using Muller's method are presented here. The first is the single precision program. The second program is in double precision and is designed to perform double precision complex arithmetic. These programs are written for use on any computer using FORTRAN IV language. They have been tested on the IBM S/360 mod. 50 computer which has a 32 bit word. However, it may be necessary to change the system functions as described below. The single precision program may be changed to double precision as described below.

After selecting the desired program, the input data should be prepared as described in section 2.

Each program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree N , where $N > 25$, certain array dimensions must be changed. These are listed in Table IX for the main program and subprograms in both single precision and double precision.

TABLE IX
 PROGRAM CHANGES FOR SOLVING POLYNOMIALS OF
 DEGREE GREATER THAN 25
 BY MULLER'S METHOD

Single PrecisionDouble Precision

Main Program

ROOT(N)
 MULT(N)
 APP(N,3)
 WORK(N+1)
 B(N+1)
 A(N+1)
 RAPP(N,3)

UROOT(N),VROOT(N)
 MULT(N)
 UAPP(N,3),VAPP(N,3)
 UWORK(N+1),VWORK(N+1)
 UB(N+1),VB(N+1)
 UA(N+1),VA(N+1)
 URAPP(N,3),VRAPP(N,3)

Subroutine BETTER

ROOT(N)
 A(N+1)
 BAPP(N,3)
 B(N+1)
 ROOTS(N)
 RAPP(N,3)
 MULT(N)

UROOT(N),VROOT(N)
 UA(N+1),VA(N+1)
 UBAPP(N,3),VBAPP(N,3)
 UB(N+1),VB(N+1)
 UROOTS(N),VROOTS(N)
 URAPP(N,3),VRAPP(N,3)
 MULT(N)

Subroutine GENAPP

APP(N,3)

APPR(N,3),APPI(N,3)

Subroutine HORNER

A(N+1)
 B(N+1)

UA(N+1),VA(N+1)
 UB(N+1),VB(N+1)

Subroutine QUAD

A(N+1)
 ROOT(N)
 MULTI(N)

UA(N+1),VA(N+1)
 UROOT(N),VROOT(N)
 MULTI(N)

Certain computers may require that the system functions of Table X be changed in the single precision and double precision programs. A

"c" denotes a complex number and an "r" denotes a real number.

TABLE X
SYSTEM FUNCTIONS USED IN MULLER'S METHOD

<u>Single Precision</u>		<u>Double Precision</u>
CABS(c)	- obtain absolute value -	DABS(r)
AIMAG(c)	- obtain imaginary part	
REAL(c)	- obtain real part	
ATAN2(r ₁ ,r ₂)	- arctangent of r ₁ /r ₂ -	DATAN2(r ₁ ,r ₂)
SQRT(r)	- square root -	DSQRT(r)
CMPLX(r ₁ ,r ₂)	- express two real numbers in complex form	
COS(r)	- cosine of angle -	DCOS(r)
SIN(r)	- sine of angle -	DSIN(r)
CSQRT(c)	- square root -	DSQRT(r)

When used on the IBM S/360 with the WATFOR compiler for FORTRAN IV, the system functions in Table X-A must be typed in a declaration statement. These also appear in the program listing. For use without the WATFOR compiler or on other machines, these system functions might have to be removed. A "c" denotes a complex number and an "r" denotes a real number.

TABLE X-A

SYSTEM FUNCTIONS IN MULLER'S METHOD TO BE TYPED
WHEN THE WATFOR COMPILER IS USED

Single PrecisionDouble Precision

Main Program and Subroutine TEST

square root - DSQRT(r)

Subroutine CALC

CMPLX(r_1, r_2)	- express in complex form	
	arctangent of r_1/r_2 -	DATAN2(r_1, r_2)
	cosine of angle -	DCOS(r)
	sine of angle -	DSIN(r)
	square root -	DSQRT(r)

Subroutine GENAPP

CMPLX(r_1, r_2)	- express in complex form	
	cosine of angle -	DCOS(r)
	sine of angle -	DSIN(r)

Subroutine ALTER

CMPLX(r_1, r_2)	- express in complex form	
	cosine of angle -	DCOS(r)
	sine of angle -	DSIN(r)
	square root -	DSQRT(r)
	arctangent of r_1/r_2 -	DATAN2(r_1, r_2)

Subroutine QUAD

CSQRT(c)	- square root -	DSQRT(r)
----------	-----------------	----------

Subroutine COMSQT

	square root -	DSQRT(r)
	absolute value -	DABS(r)

The single precision program may be converted to double precision for use on machines equipped to perform double precision complex arithmetic provided the following changes or their equivalent are made and

the system functions of Table XI are used and typed in a declaration statement where necessary. The changes presented below are those required for the IBM S/360. A "c" denotes a complex number and an "r" denotes a real number. The format statements should be changed from E-type to D-type.

In the main program and each subprogram change COMPLEX c_1, c_2, \dots to COMPLEX*16 c_1, c_2, \dots and add IMPLICIT REAL*8(A-H, O-Z).

TABLE XI

SYSTEM FUNCTIONS FOR CONVERTING SINGLE PRECISION
MULLER'S METHOD TO DOUBLE PRECISION

<u>Single Precision</u>	changed to	<u>Double Precision</u>
Main Program and Subroutines TEST and QUAD		
CABS(c)	- absolute value -	CDABS(c)
Subroutine CALC		
Add COMPLEX*8 SISC		
CABS(c)	- absolute value -	CDABS(c)
AIMAG(c)	- obtain imaginary part -	AIMAG(c) (single precision)
REAL(c)	- obtain real part -	REAL(c) (single precision)
ATAN2(r_1, r_2)	- arctangent of r_1/r_2 -	DATAN2(r_1, r_2)
SQRT(r)	- square root -	DSQRT(r)
CMPLX(r_1, r_2)	- express in complex form -	DCMPLX(r_1, r_2)
COS(r)	- cosine of angle -	DCOS(r)
SIN(r)	- sine of angle -	DSIN(r)
Subroutine GENAPP		
COS(r)	- cosine of angle -	DCOS(r)
SIN(r)	- sine of angle -	DSIN(r)
CMPLX(r_1, r_2)	- express in complex form -	DCMPLX(r_1, r_2)
Subroutine ALTER		
Add COMPLEX*8 SX2		
AIMAG(c)	- obtain imaginary part -	AIMAG(c) (single precision)

TABLE XI (Continued)

<u>Single Precision</u>	changed to	<u>Double Precision</u>
REAL(c)	- obtain real part -	REAL(c) (single precision)
CABS(c)	- absolute value -	CDABS(c)
ATAN2(r_1, r_2)	- arctangent of r_1/r_2 -	DATAN2(r_1, r_2)
COS(r)	- cosine of angle -	DCOS(r)
SIN(r)	- sine of angle -	DSIN(r)
CMPLX(r_1, r_2)	- express in complex form -	DCMPLX(r_1, r_2)
Subroutine QUAD		
CSQRT(c)	- square root -	CDSQRT(c)

2. Input Data for Muller's Method

The input data for Muller's method is identical to the input data for Newton's method as described in Appendix B, § 2 except for the variable names. The correspondence of input variable names is given in Table XII. Only one (not three) initial approximation, X_0 , is given for each root. The other two required by Muller's method are constructed within the program and are $.9X_0$ and $1.1X_0$.

3. Variables Used in Muller's Method

The definitions of the major variables used in Muller's method are given in Table XIII. For definitions of variables not listed in this table see the definitions of variables for the corresponding subroutine in Table VIII of Appendix B. The notation and symbols used here are the same as for table VIII and are described in Appendix B, § 3.

TABLE XII
CORRESPONDENCE OF NEWTON'S AND MULLER'S
INPUT DATA VARIABLES

<u>Newton's Method</u>	<u>Muller's Method</u>
Control Card	
NOPOLY	NOPOLY
N	NP
NIAP	NAPP
MAX	MAX
EPSCNV	EPS
EPSQ	EPSO
EPSMUL	EPSM
XSTART	XSTART
XEND	XEND
KCHECK	KCHECK
Coefficient Card	
A (RA)	A (UA)
A (VA)	A (VA)
Initial Approximation Card	
XZERO (RXZERO)	APP (UAPP)
XZERO (VXZERO)	APP (VAPP)
End Card	
KCHECK	KCHECK

4. Description of Program Output

The output from Muller's method is the same as that for Newton's method as described in Appendix B, § 4. Only one initial approximation, Z, (not three) is printed for each root. It is either that supplied by the user or generated by the program. The other two approximations used were 0.9Z and 1.1Z.

5. Informative and Error Messages

The output may contain informative messages printed as an aid to the user. These are:

"NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER XX."

XX is the number of the polynomial. This message is printed if no roots of the polynomial were found.

"IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(X) = YYY

DID NOT CONVERGE AFTER ZZZ ITERATIONS

THE PRESENT APPROXIMATION IS AAA"

X is the number of the root before the attempt to improve accuracy, YYY is the value of the root before attempt to improve accuracy, ZZZ is the maximum number of iterations, and AAA is the current approximation after the maximum number of iterations. This message has the same meaning as the corresponding message in Appendix B, § 5.

TABLE XIII

VARIABLES USED IN MULLER'S METHOD

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
Main Program					
NP	I	NP	I		Degree of polynomial P(X)
NROOT	I	NROOT	I		Number of distinct roots found
NOMULT	I	NOMULT	I		Number of roots (counting multiplicities)
ROOT	C	UROOT, VROOT	R		Array containing the roots
NAPP	I	NAPP	I		Number of initial approximations to be read in
APP	C	UAPP, VAPP	R		Array of initial approximations
WORK	C	UWORK, VWORK	R		Working array containing coefficients of current polynomial
B	C	UB, VB	R		Array containing coefficients of deflated polynomial
A	C	UA, VA	R		Array containing coefficients of original polynomial, P(X)
RAPP	C	URAPP, VRAPP	R		Array of initial or altered approximations for which convergence was obtained
X1	C	UX1, VX1	R		One of three current approximations to a root
X2	C	UX2, VX2	R		One of three current approximations to a root
X3	C	UX3, VX3	R		One of three current approximations to a root
PX1	C	UPX1, VPX1	R		Value of polynomial P(X) at X1
PX2	C	UPX2, VPX2	R		Value of polynomial P(X) at X2
PX3	C	UPX3, VPX3	R		Value of polynomial P(X) at X3
X4	C	UX4, VX4	R		Newest approximation (X_{n+1}) to root
PX4	C	UPX4, VPX4	R		Value of polynomial P(X) at X4
MULT	I	MULT	I		Array containing the multiplicities of each root found
ITER	I	ITER	I		Counter for iterations
I01	I	I01	I		Unit number of input device
I02	I	I02	I		Unit number of output device

TABLE XIII (Continued)

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
EPSRT	R	EPSRT	R		Number used in subroutine BETTER to generate two approximations from the one given
NOPOLY	I	NOPOLY	I		Number of the polynomial
MAX	I	MAX	I		Maximum number of iterations
EPS	R	EPS	R		Tolerance check for convergence
EPSO	R	EPSO	R		Tolerance check for zero (0)
EPSM	R	EPSM	R		Tolerance check for multiplicities
KCHECK	I	KCHECK	I		Program control, KCHECK = 1 stops execution of program
XSTART	R	XSTART	R		Magnitude at which to start generating initial approximations
XEND	R	XEND	R		Magnitude at which to end generating initial approximations
NWORK	I	NWORK	I		Degree of current deflated polynomial whose coefficients are in WORK
ITIME	I	ITIME	I		Program control
NALTER	I	NALTER	I		Number of alterations which have been performed on an initial approximation
IAPP	I	IAPP	I		Counter for number of initial approximations used
CONV	L	CONV	L		When CONV is true, convergence has been obtained
IROOT	I	IROOT	I		Number of distinct roots solved by Muller's method, i.e. not solved directly by subroutine QUAD

Subroutine HORNER

A	C	UA,VA	R	E	Array of current polynomial coefficients (to be deflated or evaluated)
NA	I	NA	I	E	Degree of polynomial to be deflated or evaluated
X	C	UX,VX	R	E	Approximation at which to evaluate or deflate the polynomial

TABLE XIII (Continued)

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition</u> <u>of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
B	C	UB,VB	R	R	Array containing the coefficients of the deflated polynomial
PX	C	UPX,VPX	R	R	Value of the polynomial at X
NUM	I	NUM	I		Number of coefficients of polynomial to be deflated
Subroutine TEST					
X3	C	UX3,VX3	R	E	Approximation to Root (old) (X_n)
X4	C	UX4,VX4	R	E	New approximation to root (X_{n+1})
CONV	L	CONV	L	R	CONV = true implies convergence has been obtained
EPS	R	EPS	R	C	Tolerance for convergence test
EPSO	R	EPSO	R	C	Tolerance check for zero (0)
DENOM	R	DENOM	R		Magnitude of new approximation, (X_{n+1})
Subroutine BETTER					
MULT	I	MULT	I	ECR	Array of multiplicities of each root
A	C	UA,VA	R	E	Array of coefficients of original undeflated polynomial
NP	I	NP	I	E	Degree of original polynomial
ROOT	C	UROOT,VROOT	R	ECR	Array of roots
NROOT	I	NROOT	I	ECR	Number of roots stored in root
BAPP	C	UBAPP,VBAPP	R	E	Array of initial approximations (old roots)
IROOT	I	IROOT	I	ECR	Number of roots solved by the iterative process (Not QUAD)
ROOTS	C	UROOTS,VROOTS	R		Temporary storage for new (better) roots
L	I	L	I		Number of roots found by BETTER
EPSRT	R	EPSRT	R	C	A small number used to generate two of the three approximations when given one
ITER	I	ITER	I	C	Counter for number of iterations

TABLE XIII (Continued)

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
B	C	UB,VB	R		Array containing coefficients of deflated polynomial
X1	C	UX1,VX1	R		One of three approximations to the root
X2	C	UX2,VX2	R		One of three approximations to the root
X3	C	UX3,VX3	R		One of three approximations to the root
PX1	C	UPX1,VPX1	R		Value of polynomial (P(X)) at X1
PX2	C	UPX2,VPX2	R		Value of polynomial (P(X)) at X2
PX3	C	UPX3,VPX3	R		Value of polynomial (P(X)) at X3
CONV	L	CONV	L		CONV = true implies convergence has been obtained
X4	C	UX4,VX4	R		Newest approximation to root
J	I	J	I		Program control -- counts the number of roots used as initial approximations
MAX	I	MAX	I	C	Maximum number of iterations permitted
IO2	I	IO2	I	C	Unit number of output device

Subroutine ALTER

X1	C	X1R,X1I	R	ECR	One of the three approximations to be altered
X2	C	X2R,X2I	R	ECR	One of the three approximations to be altered
X3	C	X3R,X3I	R	ECR	One of the three approximations to be altered
X2R	R	X2R	R		Real part of complex approximation
X2I	R	X2I	R		Imaginary part of complex approximation

Subroutine QUAD

EPST	R	EPST	R	E	Tolerance check for zero (0)
------	---	------	---	---	------------------------------

Subroutine CALC

These variables are dummy variables used for temporary storage and thus, are not defined.

MAIN PROGRAM

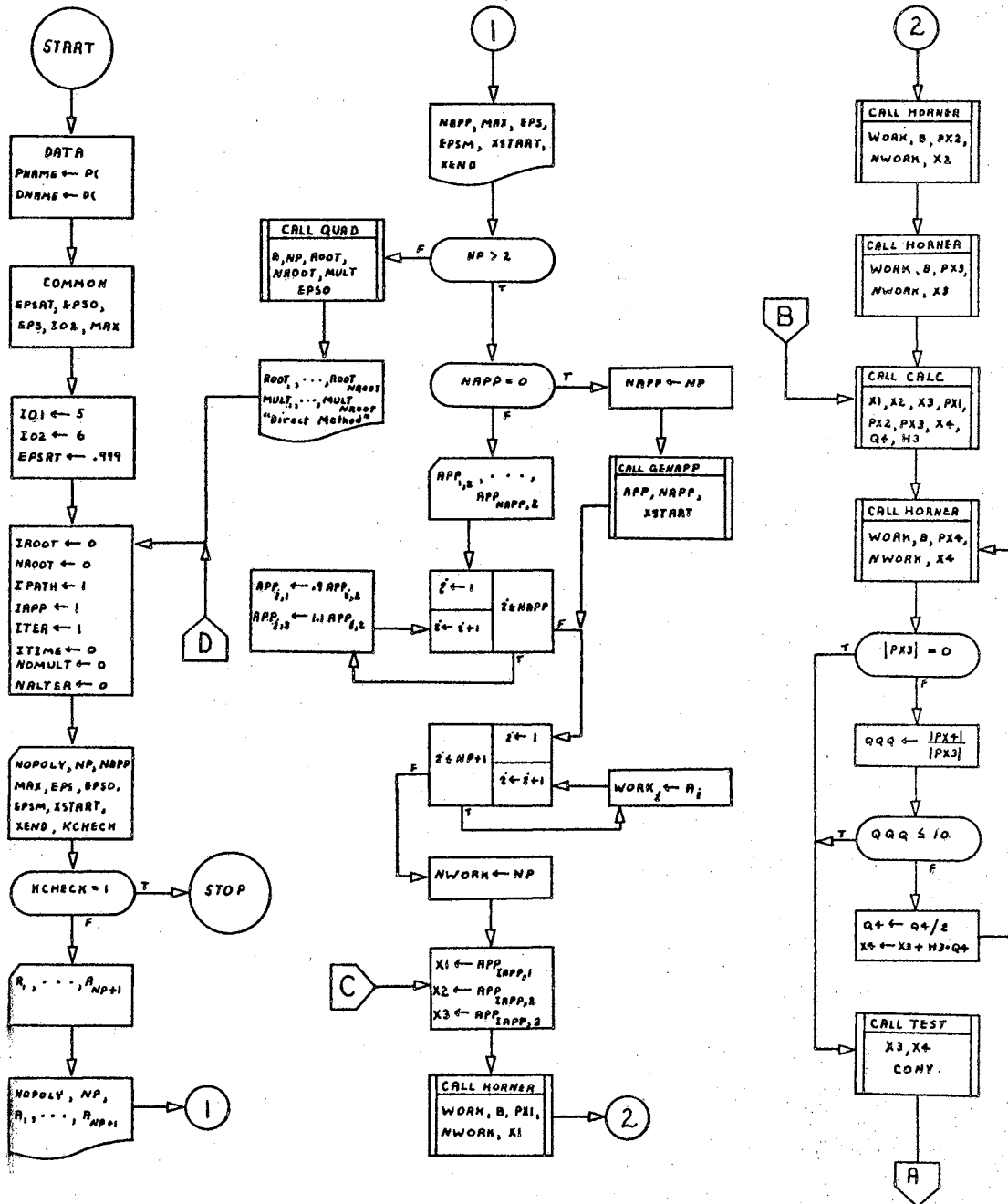


Figure 12.1. Flow Charts for Muller's Method

CALC

TEST

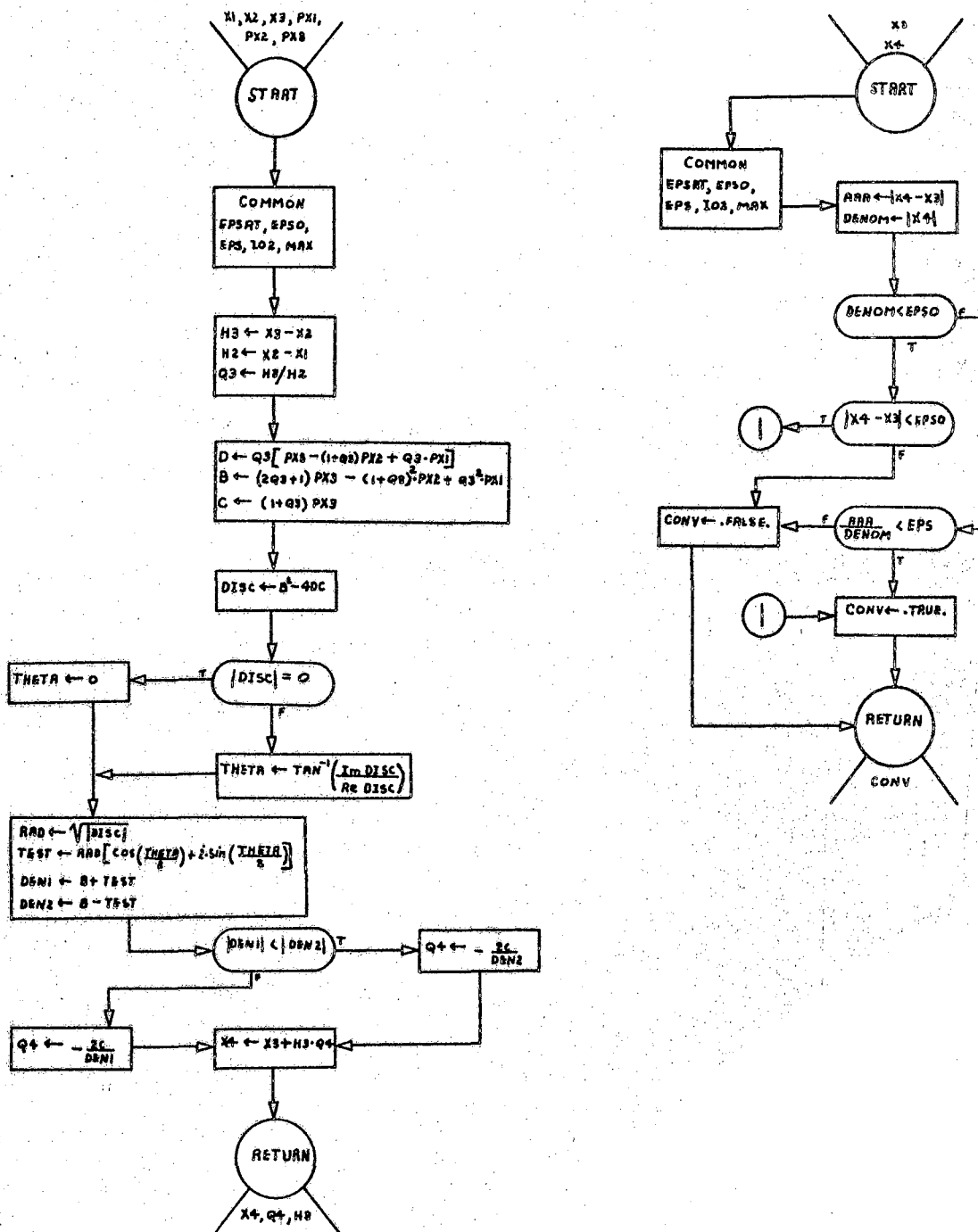


Figure 12.1. (Continued)

BETTER

HORNER

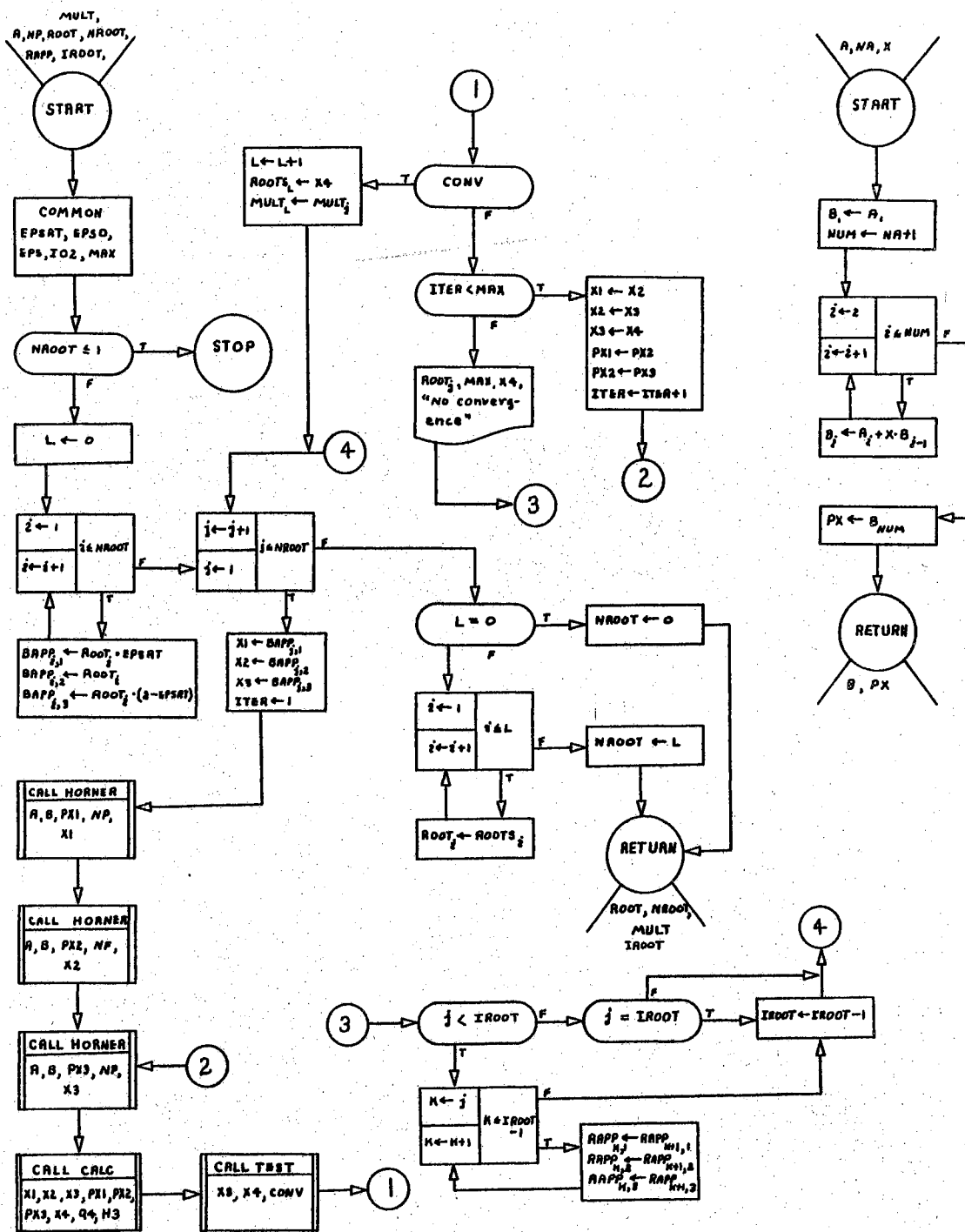
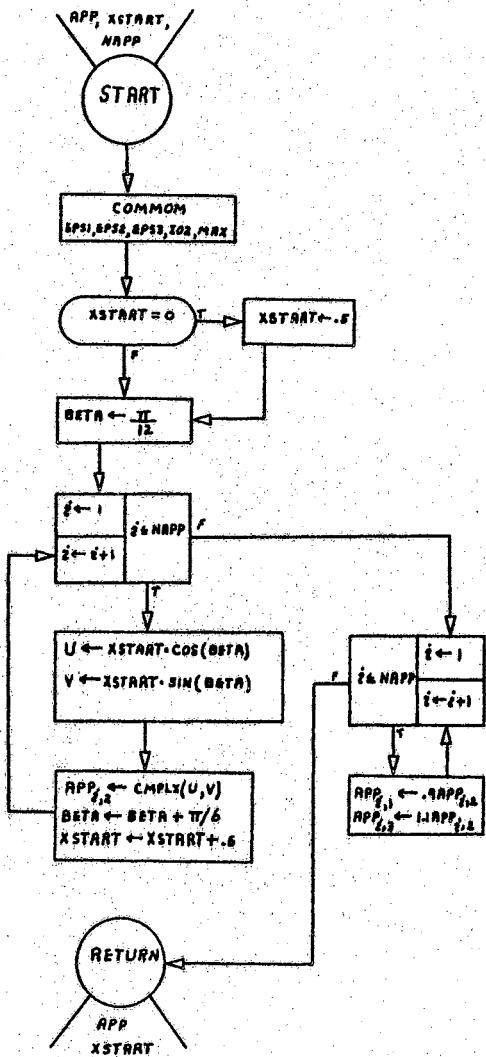


Figure 12.1. (Continued)

GENAPP



ALTER

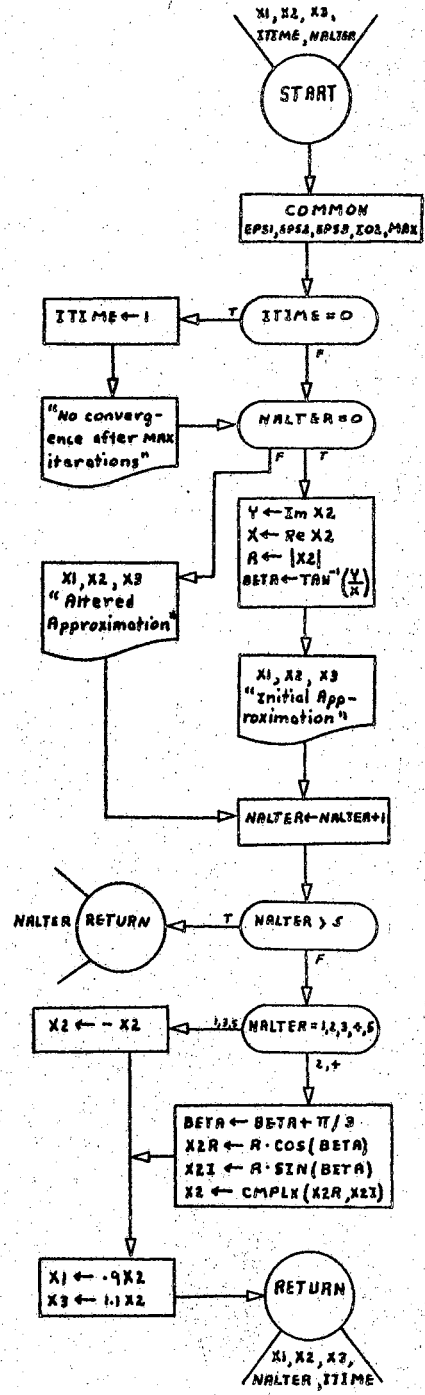
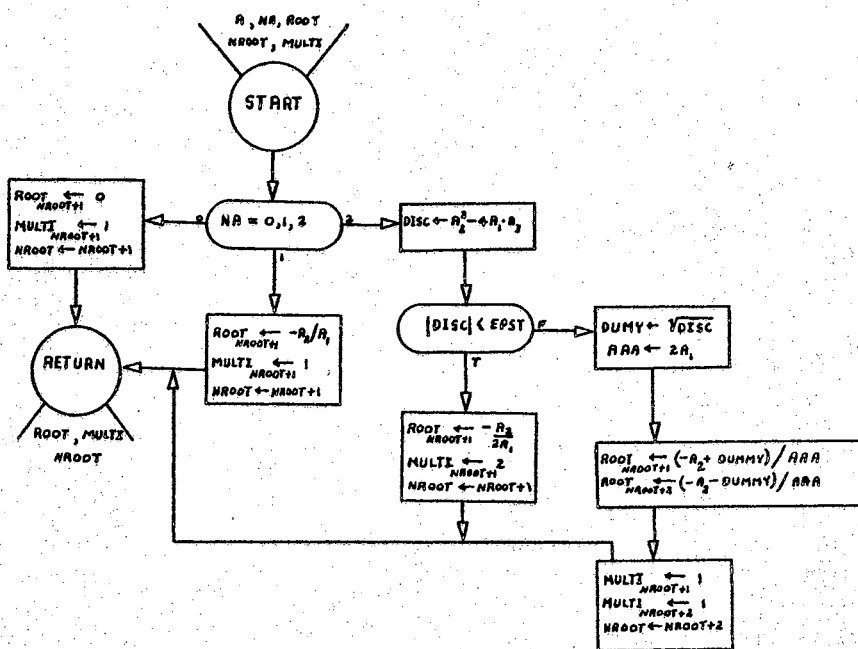


Figure 12.1. (Continued)

QUAD



COMSQT

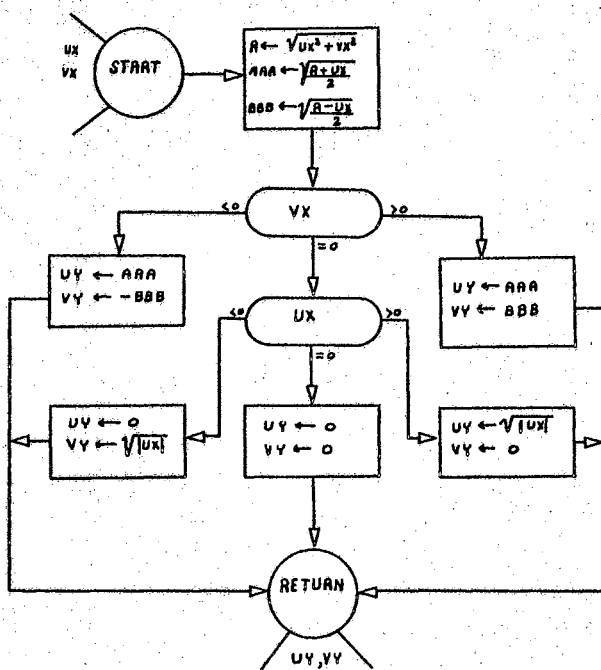


Figure 12.1. (Continued)

TABLE XIII-A

SINGLE PRECISION PROGRAM FOR MULLER'S METHOD

```

$JOB 10414
C *****
C *
C * SINGLE PRECISION PROGRAM FOR MULLER'S METHOD
C *
C *
C * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
C * POLYNOMIAL OF MAXIMUM DEGREE 25. THROUGH THREE GIVEN POINTS THE
C * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC. THE ZERO OF THE QUADRATIC
C * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.
C * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.
C *
C *****
1  COMPLEX PX3,PX2,ROOT,X1,APP,X2,WORK,X3,B,X4,A,PX1,RAPP,PX4,H3,Q4
2  DIMENSION ROOT(25),MULT(25),APP(25,3),WORK(26),B(26),A(26),RAPP(25
   1,3)
3  DATA PNAME,DNAME/2HP(,2HD(/
4  LOGICAL CONV
5  COMMON EPSRT,EPSO,EPS,IO2,MAX
6  IO1=5
7  IO2=6
8  EPSRT=0.999
9  10 NROOT=0
10  IROOT=0
11  IPATH=1
12  NALTER=0
13  ITIME=0
14  NOMULT=0
15  IAPP=1
16  ITER=1
17  READ(IO1,1000) NOPOLY,NP,NAPP,MAX,EPS,EPSO,EPSM,XSTART,XEND,KCHECK
18  IF(KCHECK.EQ.1) STOP
19  KKK=NP+1
20  READ(IO1,1010) (A(I),I=1,KKK)
21  WRITE(IO2,1020) NOPOLY,NP
22  WRITE(IO2,1035) (PNAME,I,A(I),I=1,KKK)
23  WRITE(IO2,2060)
24  WRITE(IO2,2000) NAPP
25  WRITE(IO2,2010) MAX
26  WRITE(IO2,2020) EPS
27  WRITE(IO2,2030) EPSM
28  WRITE(IO2,2040) XSTART
29  WRITE(IO2,2050) XEND
30  IF(NP.GT.2) GO TO 15
31  CALL QUAD(A,NP,ROOT,NROOT,MULT,EPSO)
32  WRITE(IO2,1037)
33  WRITE(IO2,1086) (I,ROOT(I),MULT(I),I=1,NROOT)
34  GO TO 10
35  15 IF(NAPP.NE.0) GO TO 20
36  NAPP=NP
37  CALL GENAPP(APP,NAPP,XSTART)
38  GO TO 27
39  20 READ(IO1,1030) (APP(I,2),I=1,NAPP)
40  DO 25 I=1,NAPP
41  APP(I,1)=0.9*APP(I,2)
42  25 APP(I,3)=1.1*APP(I,2)
43  27 KKK=NP+1
44  DO 30 I=1,KKK
45  30 WORK(I)=A(I)

```

TABLE XIII-A (Continued)

```

46      NWORK=NP
47      40 X1=APP(IAPP,1)
48         X2=APP(IAPP,2)
49         X3=APP(IAPP,3)
50         CALL HORNER(NWORK,WORK,X1,B,PX1)
51         CALL HORNER(NWORK,WORK,X2,B,PX2)
52         CALL HORNER(NWORK,WORK,X3,B,PX3)
53      50 CALL CALC(X1,X2,X3,PX1,PX2,PX3,X4,Q4,H3)
54      55 CALL HORNER(NWORK,WORK,X4,B,PX4)
55         AB1=CABS(PX3)
56         IF(AB1.EQ.0.0) GO TO 60
57         QQQ=CABS(PX4)/CABS(PX3)
58         IF(QQQ.LE.10.) GO TO 60
59         Q4=0.5*Q4
60         X4=X3+H3*Q4
61         GO TO 55
62      60 CALL TEST(X3,X4,CONV)
63         IF(CONV) GO TO 120
64         IF(ITER.LT.MAX) GO TO 110
65         CALL ALTER(APP(IAPP,1),APP(IAPP,2),APP(IAPP,3),NALTER,ITIME)
66         IF(NALTER.GT.5) GO TO 75
67         ITER=1
68         GO TO 40
69      75 IF(IAPP.LT.NAPP) GO TO 100
70         IF(XEND.EQ.0.0) GO TO 70
71         IF(XSTART.GT.XEND) GO TO 70
72         NAPP=NP
73         CALL GENAPP(APP,NAPP,XSTART)
74         IAPP=0
75         GO TO 100
76      70 WRITE(IO2,1090)
77         KKK=NWORK+1
78         WRITE(IO2,1035) (DNAME,J,WORK(J),J=1,KKK)
79      80 IF(NROOT.EQ.0) GO TO 90
80         WRITE(IO2,1060)
81         IF(IPATH.EQ.1) GO TO 82
82      81 IPATH=2
83         CALL BETTER(A,NP,ROOT,NROOT,RAPP,IROOT,MULT)
84         WRITE(IO2,1200)
85      82 IF(NROOT.EQ.0) GO TO 90
86         IF(IROOT.EQ.0) GO TO 85
87         WRITE(IO2,1080)
88         DO 83 I=1,IROOT
89      83 WRITE(IO2,1085) I,ROOT(I),MULT(I),RAPP(I,2)
90         IF(IROOT.LT.NROOT) GO TO 85
91         GO TO 87
92      85 KKK=IROOT+1
93         WRITE(IO2,1086) (I,ROOT(I),MULT(I),I=KKK,NROOT)
94      87 IF(IPATH.EQ.1) GO TO 81
95         GO TO 10
96      90 WRITE(IO2,1070) NOPOLY
97         GO TO 10
98      100 IAPP=IAPP+1
99         ITER=1
100         NALTER=0
101         GO TO 40
102      120 NROOT=NROOT+1
103         IROOT=NROOT
104         MULT(NROOT)=1
105         NOMULT=NOMULT+1

```

TABLE XIII-A (Continued)

```

106     ROOT(NROOT)=X4
107     RAPP(NROOT,1)=APP(IAPP,1)
108     RAPP(NROOT,2)=APP(IAPP,2)
109     RAPP(NROOT,3)=APP(IAPP,3)
110     125 IF(NOMULT.LT.NP) GO TO 130
111     GO TO 80
112     130 CALL HORNER(NWORK,WORK,X4,B,PX4)
113     NWORK=NWORK-1
114     KKK=NWORK+1
115     DO 140 I=1,KKK
116     140 WORK(I)=B(I)
117     CALL HORNER(NWORK,WORK,X4,B,PX4)
118     CCC=CABS(PX4)
119     IF(CCC.LT.EPSM) GO TO 150
120     IF(NWORK.GT.2) GO TO 75
121     IROOT=NROOT
122     CALL QUAD(WORK,NWORK,ROOT,NROOT,MULT,EPSO)
123     GO TO 80
124     150 MULT(NROOT)=MULT(NROOT)+1
125     NOMULT=NOMULT+1
126     GO TO 125
127     110 X1=X2
128     X2=X3
129     X3=X4
130     PX1=PX2
131     PX2=PX3
132     PX3=PX4
133     ITER=ITER+1
134     GO TO 50
135     1010 FORMAT(2(E30.0))
136     1020 FORMAT(1H1,1X,52HMULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOM
      1IAL/1H ,1X,18HPOLYNOMIAL NUMBER ,I2,11H OF DEGREE ,I2///1H ,1X,28H
      2THE COEFFICIENTS OF P(X) ARE//)
137     1030 FORMAT(2(E30.0))
138     1090 FORMAT(///,1X,65HCoefficients OF DEFLATED POLYNOMIAL FOR WHICH NO
      1ZEROS WERE FOUND//)
139     1080. FORMAT(///1X,13HROOTS OF P(X),37X,14HMULTIPLICITIES,11X,21HINITIAL
      1 APPROXIMATION//)
140     1070 FORMAT(///1X,42HNO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
141     1037 FORMAT(///,1X,13HZEROS OF P(X),37X,14HMULTIPLICITIES//)
142     1035 FORMAT(3X,A2,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
143     1085 FORMAT(2X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,10X,E14.
      17,3H + ,E14.7,2H I)
144     1086 FORMAT(2X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,11X,23HS
      1OLVED BY DIRECT METHOD)
145     1000 FORMAT(3(I2,1X),9X,I3,8X,3(E6.0,1X),13X,2(E7.0,1X),11)
146     2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,I2)
147     2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
148     2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,E9.2)
149     2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,E9.2)
150     2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,E9.2)
151     2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,E9.2)
152     2060 FORMAT(//1X)
153     1060 FORMAT(///35H BEFORE ATTEMPT TO IMPROVE ACCURACY)
154     1200 FORMAT(///1X,37HAFTER THE ATTEMPT TO IMPROVE ACCURACY)
155     END

```

TABLE XIII-A (Continued)

```

156      SUBROUTINE BETTER(A,NP,ROOT,NROOT,RAPP,IROOT,MULT)
C      *****
C      *
C      * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND *
C      * BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO *
C      * THE FULL, UNDEFLATED POLYNOMIAL. *
C      *
C      *****
157      COMPLEX ROOT,A,BAPP,X1,X2,X3,PX1,PX2,PX3,B,ROOTS,X4,RAPP,Q4,H3
158      LOGICAL CONV
159      DIMENSION CONV(25),A(26),BAPP(25,3),B(26),ROOTS(25),RAPP(25,3),MUL
160      IT(25)
161      COMMON EPSRT,EPSO,EPS,I02,MAX
162      IF(NROOT.LE.1) RETURN
163      L=0
164      DO 10 I=1,NROOT
165      BAPP(I,1)=ROOT(I)*EPSRT
166      BAPP(I,2)=ROOT(I)
167      10 BAPP(I,3)=ROOT(I)*(2.0-EPSRT)
168      DO 100 J=1,NROOT
169      X1=BAPP(J,1)
170      X2=BAPP(J,2)
171      X3=BAPP(J,3)
172      ITER=1
173      CALL HORNER(NP,A,X1,B,PX1)
174      CALL HORNER(NP,A,X2,B,PX2)
175      20 CALL HORNER(NP,A,X3,B,PX3)
176      CALL CALC(X1,X2,X3,PX1,PX2,PX3,X4,Q4,H3)
177      30 CALL TEST(X3,X4,CONV)
178      IF(CONV) GO TO 50
179      IF(ITER.LT.MAX) GO TO 40
180      WRITE(I02,1000) J,ROOT(J),MAX
181      WRITE(I02,1010) X4
182      IF(J.LT.IROOT) GO TO 33
183      IF(J.EQ.IROOT) GO TO 35
184      GO TO 100
185      33 KKK=IROOT-1
186      DO 34 K=J,KKK
187      RAPP(K,1)=RAPP(K+1,1)
188      RAPP(K,2)=RAPP(K+1,2)
189      34 RAPP(K,3)=RAPP(K+1,3)
190      35 IROOT=IROOT-1
191      GO TO 100
192      40 X1=X2
193      X2=X3
194      X3=X4
195      PX1=PX2
196      PX2=PX3
197      ITER=ITER+1
198      GO TO 20
199      50 L=L+1
200      ROOTS(L)=X4
201      MULT(L)=MULT(J)
202      100 CONTINUE
203      IF(L.EQ.0) GO TO 120
204      DO 110 I=1,L
205      ROOT(I)=ROOTS(I)
206      NROOT=L
207      RETURN
208      120 NROOT=0

```

TABLE XIII-A (Continued)

```

208      RETURN
209      1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,
          1E14.7,3H + ,E14.7,2H I,24H DID NOT CONVERGE AFTER ,I3,11H ITERATIO
          2NS)
210      1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,E14.7,3H + ,E14.7,2H I//)
211      END

212      SUBROUTINE CALC(X1,X2,X3,PX1,PX2,PX3,X4,Q4,H3)
C      *****
C      *
C      * GIVEN THREE APPROXIMATIONS X(N-2), X(N-1), AND X(N), SUBROUTINE CALC
C      * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF
C      * THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION
C      * X(N+1) TO THE ZERO OF THE POLYNOMIAL.
C      *
C      *****
213      COMPLEX PX3,PX2,X1,X2,X3,PX1,H3,H2,Q3,D,B,C,DISC,DEN1,DEN2,Q4,X4
214      COMPLEX TEST,CCC
215      COMPLEX CMLX
216      COMMON EPSRT,EPSO,EPS,IO2,MAX
217      H3=X3-X2
218      H2=X2-X1
219      Q3=H3/H2
220      D=Q3*(PX3-((1.0+Q3)*PX2)+(Q3*PX1))
221      B=(((2.0*Q3)+1.0)*PX3)-((1.0+Q3)*(1.0+Q3)*PX2)+(Q3*Q3*PX1)
222      C=((1.0+Q3)*PX3
223      DISC=(B*B)-(4.0*D*C)
224      AAA=CABS(DISC)
225      IF(AAA.EQ.0.0) GO TO 5
226      GO TO 7
227      5 THETA=0.0
228      GO TO 9
229      7 DISCI=AIMAG(DISC)
230      DISCR=REAL(DISC)
231      THETA=ATAN2(DISCI,DISCR)
232      9 RAD=SQRT(AAA)
233      ANGLE=THETA/2.0
234      CCC=CMLX(COS(ANGLE),SIN(ANGLE))
235      TEST=RAD*CCC
236      DEN1=B+TEST
237      DEN2=B-TEST
238      AAA=CABS(DEN1)
239      BBB=CABS(DEN2)
240      IF(AAA.LT.BBB) GO TO 10
241      IF(AAA.EQ.0.0) GO TO 60
242      Q4=(-2.0*C)/DEN1
243      GO TO 50
244      10 IF(BBB.EQ.0.0) GO TO 60
245      Q4=(-2.0*C)/DEN2
246      GO TO 50
247      50 X4=X3+(H3*Q4)
248      RETURN
249      60 Q4=(1.0,0.0)
250      GO TO 50
251      END

```

TABLE XIII-A (Continued)

```

252      SUBROUTINE GENAPP(APP,NAPP,XSTART)
C      *****
C      *
C      * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE
C      * DEGREE OF THE ORIGINAL POLYNOMIAL.
C      *
C      *****
253      COMPLEX APP
254      COMPLEX CMPLX
255      DIMENSION APP(25,3)
256      COMMON EPS1, EPS2, EPS3, IO2, MAX
257      IF(XSTART.EQ.0.0) XSTART=0.5
258      BETA=0.2617994
259      DO 10 I=1,NAPP
260      U=XSTART*COS(BETA)
261      V=XSTART*SIN(BETA)
262      APP(I,2)=CMPLX(U,V)
263      BETA=BETA+0.5235988
264      10 XSTART=XSTART+0.5
265      DO 20 I=1,NAPP
266      APP(I,1)=0.9*APP(I,2)
267      20 APP(I,3)=1.1*APP(I,2)
268      RETURN
269      END

```

TABLE XIII-A (Continued)

```

270      SUBROUTINE ALTER(X1,X2,X3,NALTER,ITIME)
C      *****
C      *
C      * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO *
C      * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
C      *
C      *****
271      COMPLEX X1,X2,X3
272      COMPLEX CPLX
273      COMMON EPS1,EPS2,EPS3,IO2,MAX
274      IF(ITIME.NE.0) GO TO 5
275      ITIME=1
276      WRITE(IO2,1010) MAX
277      5 IF(NALTER.EQ.0) GO TO 10
278      WRITE(IO2,1000) X1,X2,X3
279      GO TO 20
280      10 Y=A(MAG(X2))
281      X=REAL(X2)
282      R=CABS(X2)
283      BETA=ATAN2(Y,X)
284      WRITE(IO2,1020) X1,X2,X3
285      20 NALTER=NALTER+1
286      IF(NALTER.GT.5) RETURN
287      GO TO (30,40,30,40,30),NALTER
288      30 X2=-X2
289      GO TO 50
290      40 BETA=BETA+1.0471976
291      X2R=R*COS(BETA)
292      X2I=R*SIN(BETA)
293      X2=CMPLX(X2R,X2I)
294      50 X1=0.9*X2
295      X3=1.1*X2
296      RETURN
297      1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
          ITER ,I3,12H ITERATIONS.//)
298      1000 FORMAT(1X,5HX1 = ,E14.7,3H + ,E14.7,2H I,10X,22HALTERED APPROXIMA
          TIONS/1X,5HX2 = ,E14.7,3H + ,E14.7,2H I/1X,5HX3 = ,E14.7,3H + ,E14.
          27,2H I/)
299      1020 FORMAT(1H0,5HX1 = ,E14.7,3H + ,E14.7,2H I,10X,22HINITIAL APPROXIMA
          TIONS/1X,5HX2 = ,E14.7,3H + ,E14.7,2H I/1X,5HX3 = ,E14.7,3H + ,E14.
          2.7,2H I/)
300      END

```


TABLE XIII-A (Continued)

```

301      SUBROUTINE TEST(X3,X4,CONV)
C      *****
C      *
C      * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
C      * IMATIONS BY TESTING THE EXPRESSION
C      * ABSOLUTE VALUE OF (X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).
C      * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
C      *
C      *****
302      COMPLEX X3,X4
303      LOGICAL CONV
304      COMMON EPSRT,EPSO,EPS,IQ2,MAX
305      AAA=CABS(X4-X3)
306      DENOM=CABS(X4)
307      IF(DENOM.LT.EPSO) GO TO 20
308      IF(AAA/DENOM.LT.EPS) GO TO 10
309      5 CONV=.FALSE.
310      GO TO 100
311      10 CONV=.TRUE.
312      GO TO 100
313      20 IF(AAA.LT.EPSO) GO TO 10
314      GO TO 5
315      100 RETURN
316      END

317      SUBROUTINE HORNER(NA,A,X,B,PX)
C      *****
C      *
C      * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D.
C      * SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE
C      * FACTOR (X-D).
C      *
C      *****
318      COMPLEX X,PX,B,A
319      DIMENSION A(26),B(26)
320      B(1)=A(1)
321      NUM=NA+1
322      DO 10 I=2,NUM
323      10 B(I)=A(I)+B(I-1)*X
324      PX=B(NUM)
325      RETURN
326      END

```

TABLE XIII-A (Continued)

```

327      SUBROUTINE QUAD(A,NA,ROOT,NROOT,MULTI,EPST)
C      *****
C      *
C      * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
C      * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE *
C      * QUADRATIC IS DONE USING THE QUADRATIC FORMULA. *
C      *
C      *****
328      COMPLEX A,DISC,ROOT,DUMMY,AAA
329      COMPLEX CSQRT
330      DIMENSION A(26),ROOT(25),MULTI(25)
331      IF(NA.EQ.2) GO TO 7
332      IF(NA.EQ.1) GO TO 5
333      ROOT(NROOT+1)=0.0
334      MULTI(NROOT+1)=1
335      NROOT=NROOT+1
336      GO TO 50
337      5 ROOT(NROOT+1)=-A(2)/A(1)
338      MULTI(NROOT+1)=1
339      NROOT=NROOT+1
340      GO TO 50
341      7 DISC=A(2)*A(2)-(4.0*A(1)*A(3))
342      BBB=CABS(DISC)
343      IF(BBB.LT.EPST) GO TO 10
344      DUMMY=CSQRT(DISC)
345      AAA=2.0*A(1)
346      ROOT(NROOT+1)=(-A(2)+DUMMY)/AAA
347      ROOT(NROOT+2)=(-A(2)-DUMMY)/AAA
348      MULTI(NROOT+1)=1
349      MULTI(NROOT+2)=1
350      NROOT=NROOT+2
351      GO TO 50
352      10 ROOT(NROOT+1)=(-A(2))/(2.0*A(1))
353      MULTI(NROOT+1)=2
354      NROOT=NROOT+1
355      50 RETURN
356      END

```

\$ENTRY

TABLE XIII-B

DOUBLE PRECISION PROGRAM FOR MULLER'S METHOD

```

$JOB 10414
C *****
C *
C * DOUBLE PRECISION PROGRAM FOR MULLER'S METHOD
C *
C *
C * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
C * POLYNOMIAL OF MAXIMUM DEGREE 25. THROUGH THREE GIVEN POINTS THE
C * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC. THE ZERO OF THE QUADRATIC
C * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.
C * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.
C *
C *****
1  DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UROOT,VROOT,UX1,VX1,UAPP,VAPP
   1,UX2,VX2,UWORK,VWORK,UX3,VX3,UB,VB,UX4,VX4,UA,VA,UPX1,VPX1,URAPP,V
   2RAPP,UPX4,VPX4,EPSRT,EPSO,EPS,CCC,EPSM,UH3,VH3,UQ4,VQ4,ABPX4,ABPX3
   3,QQQ,XSTART,XEND
2  DOUBLE PRECISION DSORT
3  DIMENSION UROOT(25),VROOT(25),MULT(25),UAPP(25,3),VAPP(25,3),UWORK
   1(26),VWORK(26),UB(26),VB(26),UA(26),VA(26),URAPP(25,3),VRAPP(25,3)
4  DATA PNAME,DNAME/2HP(,2HD(/
5  LOGICAL CONV
6  COMMON EPSRT,EPSO,EPS,I02,MAX
7  I01=5
8  I02=6
9  EPSRT=0.999
10 NROOT=0
11 IROOT=0
12 IPATH=1
13 NOMULT=0
14 NALTER=0
15 ITIME=0
16 IAPP=1
17 ITER=1
18 READ(I01,1000) NOPOLY,NP,NAPP,MAX,EPS,EPSO,EPSM,XSTART,XEND,KCHECK
19 IF(KCHECK.EQ.1) STOP
20 KKK=NP+1
21 READ(I01,1010) (UA(I),VA(I),I=1,KKK)
22 WRITE(I02,1020) NOPOLY,NP
23 WRITE(I02,1035) (PNAME,I,UA(I),VA(I),I=1,KKK)
24 WRITE(I02,2060)
25 WRITE(I02,2000) NAPP
26 WRITE(I02,2010) MAX
27 WRITE(I02,2020) EPS
28 WRITE(I02,2030) EPSM
29 WRITE(I02,2040) XSTART
30 WRITE(I02,2050) XEND
31 IF(NP.GT.2) GO TO 15
32 CALL QUAD(UA,VA,NP,UROOT,VROOT,NROOT,MULT,EPSO)
33 WRITE(I02,1037)
34 WRITE(I02,1086) (I,UROOT(I),VROOT(I),MULT(I),I=1,NROOT)
35 GO TO 10
36 15 IF(NAPP.NE.0) GO TO 20
37 NAPP=NP
38 CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
39 GO TO 27
40 20 READ(I01,1030) (UAPP(I,2),VAPP(I,2),I=1,NAPP)
41 DO 25 I=1,NAPP
42 UAPP(I,1)=0.9*UAPP(I,2)

```

TABLE XIII-B (Continued)

```

43      VAPP(I,1)=0.9*VAPP(I,2)
44      UAPP(I,3)=1.1*UAPP(I,2)
45      25 VAPP(I,3)=1.1*VAPP(I,2)
46      27 KKK=NP+1
47      DO 30 I=1,KKK
48          UWORK(I)=UA(I)
49      30 VWORK(I)=VA(I)
50      NWORK=NP
51      40 UX1=UAPP(IAPP,1)
52          VX1=VAPP(IAPP,1)
53          UX2=UAPP(IAPP,2)
54          VX2=VAPP(IAPP,2)
55          UX3=UAPP(IAPP,3)
56          VX3=VAPP(IAPP,3)
57      CALL HORNER(NWORK,UWORK,VWORK,UX1,VX1,UB,VB,UPX1,VPX1)
58      CALL HORNER(NWORK,UWORK,VWORK,UX2,VX2,UB,VB,UPX2,VPX2)
59      CALL HORNER(NWORK,UWORK,VWORK,UX3,VX3,UB,VB,UPX3,VPX3)
60      50 CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
14,VX4,UQ4,VQ4,UH3,VH3)
61      60 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
62      ABPX4=DSQRT(UPX4*UPX4+VPX4*VPX4)
63      ABPX3=DSQRT(UPX3*UPX3+VPX3*VPX3)
64      IF(ABPX3.EQ.0.0) GO TO 70
65      QQQ=ABPX4/ABPX3
66      IF(QQQ.LE.10.) GO TO 70
67      UQ4=0.5*UQ4
68      VQ4=0.5*VQ4
69      UX4=UX3+(UH3*UQ4-VH3*VQ4)
70      VX4=VX3+(VH3*UQ4+UH3*VQ4)
71      GO TO 60
72      70 CALL TEST(UX3,VX3,UX4,VX4,CONV)
73      IF(CONV) GO TO 120
74      IF(ITER.LT.MAX) GO TO 110
75      CALL ALTER(UAPP(IAPP,1),VAPP(IAPP,1),UAPP(IAPP,2),VAPP(IAPP,2),UAP
IP(IAPP,3),VAPP(IAPP,3),NALTER,ITIME)
76      IF(NALTER.GT.5) GO TO 75
77      ITER=1
78      GO TO 40
79      75 IF(IAPP.LT.NAPP) GO TO 100
80      IF(XEND.EQ.0.0) GO TO 77
81      IF(XSTART.GT.XEND) GO TO 77
82      NAPP=NP
83      CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
84      IAPP=0
85      GO TO 100
86      77 WRITE(IO2,1090)
87      KKK=NWORK+1
88      WRITE(IO2,1035) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
89      80 IF(NROOT.EQ.0) GO TO 90
90      WRITE(IO2,1060)
91      IF(IPATH.EQ.1) GO TO 82
92      81 IPATH=2
93      CALL BETTER(UA,VA,NP,URROOT,VROOT,NROOT,URAPP,VAPP,IROOT,MULT)
94      WRITE(IO2,1200)
95      82 IF(NROOT.EQ.0) GO TO 90
96      IF(IROOT.EQ.0) GO TO 85
97      WRITE(IO2,1080)
98      DO 55 I=1,IROOT
99      55 WRITE(IO2,1085) I,URROOT(I),VROOT(I),MULT(I),URAPP(I,2),VAPP(I,2)
100     IF(IROOT.LT.NROOT) GO TO 85

```

TABLE XIII-B (Continued)

```

101      GO TO 87
102      85 KKK=IROOT+1
103      WRITE(102,1086) (I,UROOT(I),VROOT(I),MULT(I),I=KKK,NROOT)
104      87 IF(IPATH.EQ.1) GO TO 81
105      GO TO 10
106      90 WRITE(102,1070) NOPOLY
107      GO TO 10
108      100 IAPP=IAPP+1
109      ITER=1
110      NALTER=0
111      GO TO 40
112      120 NROOT=NROOT+1
113      IROOT=NROOT
114      MULT(NROOT)=1
115      NOMULT=NOMULT+1
116      UROOT(NROOT)=UX4
117      VROOT(NROOT)=VX4
118      URAPP(NROOT,1)=UAPP(IAPP,1)
119      VRAPP(NROOT,1)=VAPP(IAPP,1)
120      URAPP(NROOT,2)=UAPP(IAPP,2)
121      VRAPP(NROOT,2)=VAPP(IAPP,2)
122      URAPP(NROOT,3)=UAPP(IAPP,3)
123      VRAPP(NROOT,3)=VAPP(IAPP,3)
124      125 IF(NOMULT.LT.NP) GO TO 130
125      GO TO 80
126      130 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
127      NWORK=NWORK-1
128      KKK=NWORK+1
129      DO 140 I=1,KKK
130      UWORK(I)=UB(I)
131      140 VWORK(I)=VB(I)
132      CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
133      CCC=DSQRT(UPX4*UPX4+VPX4*VPX4)
134      IF(CCC.LT.EPSM) GO TO 150
135      IF(NWORK.GT.2) GO TO 75
136      IROOT=NROOT
137      CALL QUAD(UWORK,VWORK,NWORK,UROOT,VROOT,NROOT,MULT,EPSO)
138      GO TO 80
139      150 MULT(NROOT)=MULT(NROOT)+1
140      NOMULT=NOMULT+1
141      GO TO 125
142      110 UX1=UX2
143      VX1=VX2
144      UX2=UX3
145      VX2=VX3
146      UX3=UX4
147      VX3=VX4
148      UPX1=UPX2
149      VPX1=VPX2
150      UPX2=UPX3
151      VPX2=VPX3
152      UPX3=UPX4
153      VPX3=VPX4
154      ITER=ITER+1
155      GO TO 50
156      1010 FORMAT(2D30.0)
157      1020 FORMAT(11H,1X,52HMULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOM
158      1IAL/1H ,1X,18HPOLYNOMIAL NUMBER ,I2,11H OF DEGREE ,I2///1H ,1X,28H
158      2THE COEFFICIENTS OF P(X) ARE//)
1030 FORMAT(2D30.0)

```

TABLE XIII-B (Continued)

```

159 1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
      1ZEROS WERE FOUND//)
160 1080 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
      1 APPROXIMATION//)
161 1070 FORMAT(//,43H NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
162 1086 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,8X,I2,9X,23HS
      1OLVED BY DIRECT METHOD)
163 1037 FORMAT(///,1X,13HZEROS OF P(X),51X,14HMULTIPLICITIES//)
164 1035 FORMAT(3X,A2,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
165 1085 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,8X,I2,8X,D23.
      116,3H + ,D23.16,2H I)
166 1000 FORMAT(3(I2,1X),9X,I3,8X,3(D6.0,1X),13X,2(D7.0,1X),I1)
167 1060 FORMAT(///35H BEFORE ATTEMPT TO IMPROVE ACCURACY)
168 1200 FORMAT(///1X,37HAFTER THE ATTEMPT TO IMPROVE ACCURACY)
169 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,I2)
170 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
171 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
172 2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)
173 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
174 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
175 2060 FORMAT(//1X)
176      END

```

TABLE XIII-B (Continued)

```

177      SUBROUTINE ALTER(X1R,X1I,X2R,X2I,X3R,X3I,NALTER,ITIME)
C      *****
C      *
C      * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO
C      * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.
C      *
C      *****
178      DOUBLE PRECISION X1R,X1I,X2R,X2I,X3R,X3I,EPS1,EPS2,EPS3,R,BETA
179      DOUBLE PRECISION DCOS,DSIN
180      DOUBLE PRECISION DSQRT
181      DOUBLE PRECISION DATAN2
182      COMMON EPS1,EPS2,EPS3,IO2,MAX
183      IF(ITIME.NE.0) GO TO 5
184      ITIME=1
185      WRITE(IO2,1010) MAX
186      5 IF(NALTER.EQ.0) GO TO 10
187      WRITE(IO2,1000) X1R,X1I,X2R,X2I,X3R,X3I
188      GO TO 20
189      10 R=DSQRT(X2R*X2R+X2I*X2I)
190      BETA=DATAN2(X2I,X2R)
191      WRITE(IO2,1020) X1R,X1I,X2R,X2I,X3R,X3I
192      20 NALTER=NALTER+1
193      IF(NALTER.GT.5) RETURN
194      GO TO (30,40,30,40,30),NALTER
195      30 X2R=-X2R
196      X2I=-X2I
197      GO TO 50
198      40 BETA=BETA+1.0471976
199      X2R=R*DCOS(BETA)
200      X2I=R*DSIN(BETA)
201      50 X1R=0.9*X2R
202      X1I=0.9*X2I
203      X3R=1.1*X2R
204      X3I=1.1*X2I
205      RETURN
206      1000 FORMAT(1X,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22HALTERED APPROXIM
LATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
2,D23.16,2H I/)
207      1020 FORMAT(1H0,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22INITIAL APPROXI
MATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
2 ,D23.16,2H I/)
208      1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
ITER ,I3,12H ITERATIONS.//)
209      END

```

TABLE XIII-B (Continued)

```

210      SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
C      *****
C      *
C      * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE *
C      * DEGREE OF THE ORIGINAL POLYNOMIAL. *
C      *
C      *****
211      DOUBLE PRECISION APPR,APPI,XSTART,EPS1,EPS2,EPS3,BETA
212      DOUBLE PRECISION DCOS,DSIN
213      DIMENSION APPR(25,3),APPI(25,3)
214      COMMON EPS1,EPS2,EPS3,IO2,MAX
215      IF(XSTART.EQ.0.0) XSTART=0.5
216      BETA=0.2617994
217      DO 10 I=1,NAPP
218      APPR(I,2)=XSTART*DCOS(BETA)
219      APPI(I,2)=XSTART*DSIN(BETA)
220      BETA=BETA+0.5235988
221      10 XSTART=XSTART+0.5
222      DO 20 I=1,NAPP
223      APPR(I,1)=0.9*APPR(I,2)
224      APPI(I,1)=0.9*APPI(I,2)
225      APPR(I,3)=1.1*APPR(I,2)
226      20 APPI(I,3)=1.1*APPI(I,2)
227      RETURN
228      END

```


TABLE XIII-B (Continued)

```

229 SUBROUTINE BETTER(UA,VA, NP, UROOT, VROOT, NROOT, URAPP, VRAPP, IROOT, MUL
      1T)
C *****
C *
C * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND *
C * BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO *
C * THE FULL, UNDEFLATED POLYNOMIAL. *
C *
C *****
230 DOUBLE PRECISION UROOT, VROOT, UA, VA, UBAPP, VBAPP, UX1, VX1, UX2, VX2, UX3
      1, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, VPX3, UB, VB, UROOTS, VROOTS, EPSRT, UX4, V
      2X4, URAPP, VRAPP, EPSO, EPS, UQ4, VQ4, UH3, VH3
231 LOGICAL CONV
232 DIMENSION UROOT(25), VROOT(25), UA(26), VA(26), UBAPP(25,3), VBAPP(25,3
      1), UB(26), VB(26), UROOTS(25), VROOTS(25), URAPP(25,3), VRAPP(25,3), MULT
      3(25)
233 COMMON EPSRT, EPSO, EPS, IO2, MAX
234 IF(NROOT.LE.1) RETURN
235 L=0
236 DO 10 I=1, NROOT
237   UBAPP(I,1)=UROOT(I)*EPSRT
238   VBAPP(I,1)=VROOT(I)*EPSRT
239   UBAPP(I,2)=UROOT(I)
240   VBAPP(I,2)=VROOT(I)
241   UBAPP(I,3)=UROOT(I)*(2.0-EPSRT)
242 10  VBAPP(I,3)=VROOT(I)*(2.0-EPSRT)
243   DO 100 J=1, NROOT
244     UX1=UBAPP(J,1)
245     VX1=VBAPP(J,1)
246     UX2=UBAPP(J,2)
247     VX2=VBAPP(J,2)
248     UX3=UBAPP(J,3)
249     VX3=VBAPP(J,3)
250     ITER=1
251     CALL HORNER(NP, UA, VA, UX1, VX1, UB, VB, UPX1, VPX1)
252     CALL HORNER(NP, UA, VA, UX2, VX2, UB, VB, UPX2, VPX2)
253 20  CALL HORNER(NP, UA, VA, UX3, VX3, UB, VB, UPX3, VPX3)
254     CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, VPX3, UX
      14, VX4, UQ4, VQ4, UH3, VH3)
255 30  CALL TEST(UX3, VX3, UX4, VX4, CONV)
256     IF(CONV) GO TO 50
257     IF(ITER.LT.MAX) GO TO 40
258     WRITE(IO2,1000) J, UROOT(J), VROOT(J), MAX
259     WRITE(IO2,1010) UX4, VX4
260     IF(J.LT.IROOT) GO TO 33
261     IF(J.EQ.IROOT) GO TO 35
262     GO TO 100
263 33  KKK=IROOT-1
264     DO 34 K=J, KKK
265       URAPP(K,1)=URAPP(K+1,1)
266       VRAPP(K,1)=VRAPP(K+1,1)
267       URAPP(K,2)=URAPP(K+1,2)
268       VRAPP(K,2)=VRAPP(K+1,2)
269       URAPP(K,3)=URAPP(K+1,3)
270 34  VRAPP(K,3)=VRAPP(K+1,3)
271 35  IROOT=IROOT-1
272     GO TO 100
273 40  UX1=UX2
274     VX1=VX2
275     UX2=UX3

```

TABLE XIII-B (Continued)

```

276      VX2=VX3
277      UX3=UX4
278      VX3=VX4
279      UPX1=UPX2
280      VPX1=VPX2
281      UPX2=UPX3
282      VPX2=VPX3
283      ITER=ITER+1
284      GO TO 20
285  50  L=L+1
286      UROOTS(L)=UX4
287      VROOTS(L)=VX4
288      MULT(L)=MULT(J)
289  100  CONTINUE
290      IF(L.EQ.0) GO TO 120
291      DO 110 I=1,L
292      UROOT(I)=UROOTS(I)
293  110  VROOT(I)=VROOTS(I)
294      NROOT=L
295      RETURN
296  120  NROOT=0
297      RETURN
298  1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(I,I2,4H) = ,
      1D23.16,3H + ,D23.16,2H I/24H DID NOT CONVERGE AFTER ,I3,11H ITERAT
      2IONS)
299  1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,D23.16,3H + ,D23.16,2H I/
      1/)
300      END

```

TABLE XIII-B (Continued)

```

301      SUBROUTINE CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,V
          1PX3,UX4,VX4,UQ4,VQ4,UH3,VH3)
C      *****
C      *
C      * GIVEN THREE APPROXIMATIONS X(N-2), X(N-1), AND X(N), SUBROUTINE CALC
C      * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF
C      * THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION
C      * X(N+1) TO THE ZERO OF THE POLYNOMIAL.
C      *
C      *****
302      DOUBLE PRECISION ARG1,ARG2
303      DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UX1,VX1,UX2,VX2,UX3,VX3,UPX1,
          1VPX1,UH3,VH3,UH2,VH2,UQ3,VQ3,UD,VD,UB,VB,UC,VC,UDISC,VDISC,UCCC,VC
          2CC,UDEN1,VDEN1,UDEN2,VDEN2,UQ4,VQ4,UX4,VX4,EPSRT,EPSO,EPS,UDDD,VDD
          3D,AAA,BBB,RAD,UAAA,VAAA,UBBB,VBBB
304      DOUBLE PRECISION THETA,ANGLE,UTEST,VTEST
305      DOUBLE PRECISION DATAN2,DCOS,DSIN,DSQRT
306      COMMON EPSRT,EPSO,EPS,IO2,MAX
307      UH3=UX3-UX2
308      VH3=VX3-VX2
309      UH2=UX2-UX1
310      VH2=VX2-VX1
311      BBB=UH2*UH2+VH2*VH2
312      UQ3=(UH3*UH2+VH3*VH2)/BBB
313      VQ3=(VH3*UH2-UH3*VH2)/BBB
314      UDDD=1.0+UQ3
315      VDDD=VQ3
316      UD=(UPX3-(UDDD*UPX2-VDDD*VPX2))+{UQ3*UPX1-VQ3*VPX1}
317      VD=(VPX3-(VDDD*UPX2+UDDD*VPX2))+{VQ3*UPX1+UQ3*VPX1}
318      UAAA=2.0*UQ3
319      VAAA=2.0*VQ3
320      UAAA=UAAA+1.0
321      UBBB=UDDD*UDDD-VDDD*VDDD
322      VBBB=VDDD*UDDD+UDDD*VDDD
323      UCCC=UQ3*UQ3-VQ3*VQ3
324      VCCC=VQ3*UQ3+UQ3*VQ3
325      UB=({UAAA*UPX3-VAAA*VPX3}-{UBBB*UPX2-VBBB*VPX2}))+{UCCC*UPX1-VCCC*V
          1PX1}
326      VB=({VAAA*UPX3+UAAA*VPX3}-{VBBB*UPX2+UBBB*VPX2}))+{VCCC*UPX1+UCCC*V
          1PX1}
327      UC=UDDD*UPX3-VDDD*VPX3
328      VC=VDDD*UPX3+UDDD*VPX3
329      UDISC=(UB*UB-VB*VB)-(4.0*(UD*UC-VD*VC))
330      VDISC=(2.0*(VB*UB))-(4.0*(VD*UC+UD*VC))
331      AAA=DSQRT(UDISC*UDISC+VDISC*VDISC)
332      IF(AAA.EQ.0.0) GO TO 5
333      GO TO 7
334      5 THETA=0.0
335      GO TO 9
336      7 THETA=DATAN2(VDISC,UDISC)
337      9 RAD=DSQRT(AAA)
338      ANGLE=THETA/2.0
339      UTEST=RAD*DCOS(ANGLE)
340      VTEST=RAD*DSIN(ANGLE)
341      UDEN1=UB+UTEST
342      VDEN1=VB+VTEST
343      UDEN2=UB-UTEST
344      VDEN2=VB-VTEST
345      ARG1=UDEN1*UDEN1+VDEN1*VDEN1
346      ARG2=UDEN2*UDEN2+VDEN2*VDEN2

```

TABLE XIII-B (Continued)

```

347      AAA=DSQRT(ARG1)
348      BBB=DSQRT(ARG2)
349      IF(AAA.LT.BBB) GO TO 10
350      IF(AAA.EQ.0.0) GO TO 60
351      UAAA=-2.0*UC
352      VAAA=-2.0*VC
353      UQ4=(UAAA*UDEN1+VAAA*VDEN1)/ARG1
354      VQ4=(VAAA*UDEN1-UAAA*VDEN1)/ARG1
355      GO TO 50
356 10 IF(BBB.EQ.0.0) GO TO 60
357      UAAA=-2.0*UC
358      VAAA=-2.0*VC
359      UQ4=(UAAA*UDEN2+VAAA*VDEN2)/ARG2
360      VQ4=(VAAA*UDEN2-UAAA*VDEN2)/ARG2
361      GO TO 50
362 50 UX4=UX3+(UH3*UQ4-VH3*VQ4)
363      VX4=VX3+(VH3*UQ4+UH3*VQ4)
364      RETURN
365 60 UQ4=1.0
366      VQ4=0.0
367      GO TO 50
368      END

369      SUBROUTINE TEST(UX3,VX3,UX4,VX4,CONV)
C      *****
C      *
C      * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
C      * IMATIONS BY TESTING THE EXPRESSION
C      * ABSOLUTE VALUE OF {X(N+1)-X(N)}/ABSOLUTE VALUE OF X(N+1).
C      * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
C      *
C      *****
370      DOUBLE PRECISION DSQRT
371      DOUBLE PRECISION UX3,VX3,UX4,VX4,EPSRT,EPS0,EPS,AAA,UDUMMY,VDUMMY,
      DENOM
372      LOGICAL CONV
373      COMMON EPSRT,EPS0,EPS,IO2,MAX
374      UDUMMY=UX4-UX3
375      VDUMMY=VX4-VX3
376      AAA=DSQRT(UDUMMY*UDUMMY+VDUMMY*VDUMMY)
377      DENOM=DSQRT(UX4*UX4+VX4*VX4)
378      IF(DENOM.LT.EPS0) GO TO 20
379      IF(AAA/DENOM.LT.EPS) GO TO 10
380      5 CONV=.FALSE.
381      GO TO 100
382      10 CONV=.TRUE.
383      GO TO 100
384      20 IF(AAA.LT.EPS0) GO TO 10
385      GO TO 5
386 100 RETURN
387      END

```

TABLE XIII-B (Continued)

```

388      SUBROUTINE HORNER(NA,UA,VA,UX,VX,UB,VB,UPX,VPX)
C      *****
C      *
C      * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D. *
C      * SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE *
C      * FACTOR (X-D). *
C      *
C      *****
389      DOUBLE PRECISION UX,VX,UPX,VPX,UB,VB,UA,VA
390      DIMENSION UA(26),VA(26),UB(26),VB(26)
391      UB(1)=UA(1)
392      VB(1)=VA(1)
393      NUM=NA+1
394      DO 10 I=2,NUM
395      UB(I)=UA(I)+(UB(I-1)*UX-VB(I-1)*VX)
396      10 VB(I)=VA(I)+(VB(I-1)*UX+UB(I-1)*VX)
397      UPX=UB(NUM)
398      VPX=VB(NUM)
399      RETURN
400      END

```

TABLE XIII-B (Continued)

```

401 SUBROUTINE QUAD(UA,VA,NA,UROOT,VROOT,NROOT,MULTI,EPST)
C *****
C *
C * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
C * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE *
C * QUADRATIC IS DONE USING THE QUADRATIC FORMULA. *
C *
C *****
402 DOUBLE PRECISION UA,VA,UROOT,VROOT,BBB,UAAA,VAAA,UDISC,VDISC,UDUMM
1Y,VDUMMY,RDUMMY,SDUMMY,EPST,UBBB,VBBB
403 DOUBLE PRECISION DSQRT
404 DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),MULTI(25)
405 IF(NA.EQ.2) GO TO 7
406 IF(NA.EQ.1) GO TO 5
407 UROOT(NROOT+1)=0.0
408 VROOT(NROOT+1)=0.0
409 MULTI(NROOT+1)=1
410 NROOT=NROOT+1
411 GO TO 50
412 5 BBB=UA(1)*UA(1)+VA(1)*VA(1)
413 UROOT(NROOT+1)=(-UA(2)*UA(1)-VA(2)*VA(1))/BBB
414 VROOT(NROOT+1)=(-VA(2)*UA(1)+UA(2)*VA(1))/BBB
415 MULTI(NROOT+1)=1
416 NROOT=NROOT+1
417 GO TO 50
418 7 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-4.0*(UA(1)*UA(3)-VA(1)*VA(3))
419 VDISC=(VA(2)*UA(2)+UA(2)*VA(2))-4.0*(VA(1)*UA(3)+UA(1)*VA(3))
420 BBB=DSQRT(UDISC*UDISC+VDISC*VDISC)
421 IF(BBB.LT.EPST) GO TO 10
422 CALL COMSQT(UDISC,VDISC,UDUMMY,VDUMMY)
423 UBBB=-UA(2)+UDUMMY
424 VBBB=-VA(2)+VDUMMY
425 RDUMMY=-UA(2)-UDUMMY
426 SDUMMY=-VA(2)-VDUMMY
427 UAAA=2.0*UA(1)
428 VAAA=2.0*VA(1)
429 BBB=UAAA*UAAA+VAAA*VAAA
430 UROOT(NROOT+1)=(UBBB*UAAA+VBBB*VAAA)/BBB
431 VROOT(NROOT+1)=(VBBB*UAAA-UBBB*VAAA)/BBB
432 UROOT(NROOT+2)=(RDUMMY*UAAA+SDUMMY*VAAA)/BBB
433 VROOT(NROOT+2)=(SDUMMY*UAAA-RDUMMY*VAAA)/BBB
434 MULTI(NROOT+1)=1
435 MULTI(NROOT+2)=1
436 NROOT=NROOT+2
437 GO TO 50
438 10 UAAA=2.0*UA(1)
439 VAAA=2.0*VA(1)
440 BBB=UAAA*UAAA+VAAA*VAAA
441 UROOT(NROOT+1)=(-UA(2)*UAAA-VA(2)*VAAA)/BBB
442 VROOT(NROOT+1)=(-VA(2)*UAAA+UA(2)*VAAA)/BBB
443 MULTI(NROOT+1)=2
444 NROOT=NROOT+1
445 50 RETURN
446 END

```

TABLE XIII-B (Continued)

```

447      SUBROUTINE COMSQT(UX,VX,UY,VY)
C      *****
C      *
C      * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
C      *
C      *****
448      DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
449      DOUBLE PRECISION DSQRT,DABS
450      R=DSQRT(UX*UX+VX*VX)
451      AAA=DSQRT(DABS((R+UX)/2.0))
452      BBB=DSQRT(DABS((R-UX)/2.0))
453      IF(VX) 10,20,30
454      10  UY=AAA
455          VY=-1.0*BBB
456          GO TO 100
457      20  IF(UX) 40,50,60
458      30  UY=AAA
459          VY=BBB
460          GO TO 100
461      40  DUMMY=DABS(UX)
462          UY=0.0
463          VY=DSQRT(DUMMY)
464          GO TO 100
465      50  UY=0.0
466          VY=0.0
467          GO TO 100
468      60  DUMMY=DABS(UX)
469          UY=DSQRT(DUMMY)
470          VY=0.0
471      100 RETURN
472      END

```

\$ENTRY

APPENDIX D

G.C.D. - NEWTON'S METHOD

1. Use of the Programs

Two programs using the greatest common divisor method supported by Newton's method are presented here. The first is the single precision program. The second program is in double precision and is designed to perform double precision complex arithmetic. These programs are written for use on any computer using FORTRAN IV language. They have been tested on the IBM S/360 mod. 50 computer which has a 32 bit word. However, it may be necessary to change the system functions as described below. The single precision program may be changed to double precision as described below.

After selecting the desired program, the input data should be prepared as described in section 2.

Each program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree N , where $N > 25$, certain array dimensions must be changed. These are listed in Table XIV for the main program and subprograms in both single precision and double precision.

TABLE XIV

PROGRAM CHANGES FOR SOLVING POLYNOMIALS
OF DEGREE GREATER THAN 25 BY G.C.D. -
NEWTON'S METHOD

Single PrecisionDouble Precision

Main Program

P(N+1)	UP(N+1),VP(N+1)
OIAPP(N)	UOIAPP(N),VOIAPP(N)
ROOT(N)	UROOT(N),VROOT(N)
MULTI(N)	MULTI(N)
DP(N+1)	UDP(N+1),VDP(N+1)
RK(N+1)	URK(N+1),VRK(N+1)
RIAPP(N)	URIAPP(N),VRIAPP(N)
H(N+1)	UH(N+1),VH(N+1)
DPOLY(N+1)	UDPOLY(N+1),VDPOLY(N+1)

Subroutine METHOD

See main program of Table I in Appendix B.

Subroutines GENAPP, HORNER, BETTER, and QUAD

See corresponding part of Table I in Appendix B.

Subroutine DERIV

A(N+1)	UA(N+1),VA(N+1)
B(N+1)	UB(N+1),VB(N+1)

Subroutine DIVIDE

F(N+1)	UF(N+1),VF(N+1)
G(N+1)	UG(N+1),VG(N+1)
H(N+1)	UH(N+1),VH(N+1)
C(N+1)	UC(N+1),VC(N+1)

Subroutine GCD

R1(N+1)	UR1(N+1),VR1(N+1)
R2(N+1)	UR2(N+1),VR2(N+1)
T(N+1)	UT(N+1),VT(N+1)
RK(N+1)	URK(N+1),VRK(N+1)
P(N+1)	UP(N+1),VP(N+1)
DP(N+1)	UDP(N+1),VDP(N+1)

TABLE XIV (Continued)

<u>Single Precision</u>	<u>Double Precision</u>
Subroutine MULTIP	
A(N+1)	UA(N+1), VA(N+1)
ROOT(N)	UROOT(N), VROOT(N)
WORK(N+1)	UWORK(N+1), VWORK(N+1)
MULTI(N)	MULTI(N)
B(N+1)	UB(N+1), VB(N+1)
C(N+1)	UC(N+1), VC(N+1)

Certain computers may require that the system functions of Table II in Appendix B be changed in the single precision and double precision programs.

When used on the IBM S/360 with the WATFOR compiler for FORTRAN IV, the system functions in Table XIV-A must be typed in a declaration statement. These also appear in the program listing. For use without the WATFOR compiler or on other computers, these system functions might have to be removed. A "c" denotes a complex number and an "r" denotes a real number. For subroutines not listed, see the corresponding subroutine of Table II-A in Appendix B.

TABLE XIV-A

SYSTEM FUNCTIONS IN THE G.C.D. - NEWTON'S METHOD
TO BE TYPED WHEN THE WATFOR COMPILER IS USED

<u>Single Precision</u>	<u>Double Precision</u>
Subroutines METHOD, GCD, and MULTIP	
Square root -	DSQRT(r)

The single precision program may be converted to double precision for use on machines equipped to perform double precision complex arithmetic provided the following changes or their equivalent are made and the system functions of Table XV are used and typed in a declaration statement where necessary. The changes presented below are those required for the IBM S/360. A "c" denotes a complex number and an "r" denotes a real number. The format statements should be changed from E-type to D-type.

In the main program and each subprogram change COMPLEX c_1, c_2, \dots to COMPLEX*16 c_1, c_2, \dots and add IMPLICIT REAL*8(A-H,O-Z).

TABLE XV

SYSTEM FUNCTIONS FOR CONVERTING SINGLE PRECISION G.C.D. -
NEWTON'S METHOD TO DOUBLE PRECISION

Single PrecisionDouble Precision

Subroutines METHOD, ALTER, NEWTON, CHECK,
BETTER, QUAD, GCD, and MULTIP

CABS(c)

- absolute value -

CDABS(c)

Subroutines GENAPP, ALTER, and QUAD

See corresponding part of Table III in Appendix B.

2. Input Data for G.C.D. - Newton's Method

The input data for the G.C.D. - Newton's method is prepared as described in Appendix B, § 2 except for some of the variable names and

one added item to the control card. The control card is described in Table XVI and illustrated in Figure 13. The correspondence of variable names for the coefficient data, initial approximation data, and end card are given in Table XVII.

TABLE XVI

CONTROL DATA FOR G.C.D. - NEWTON'S METHOD

<u>Variable Name</u>	<u>Card Columns</u>	<u>Description</u>
NOPOLY	c.c. 1-2	Number of the polynomial. Integer. Right justified.
NP	c.c. 4-5	Degree of the polynomial. Integer. Right justified.
NIAP	c.c. 7-8	Number of initial approximations to be read. Integer. Right justified. If no initial approximations are given, leave blank.
MAX	c.c. 19-21	Maximum number of iterations. Integer. Right justified. 200 is recommended.
EPSG	c.c. 23-28	Test for zero in subroutine GCD. Real. Right justify. 1.E-02 (1.D-03) is recommended.
EPSCNV	c.c. 30-35	Convergence requirement. Real. Right justify. 1.E-05 (1.D-10) is recommended.
EPSQ	c.c. 37-42	Test for zero in subroutine QUAD. Real. Right justify. 1.E-10 (1.D-20) is recommended.

TABLE XVI (Continued)

<u>Variable Name</u>	<u>Card Columns</u>	<u>Description</u>
EPSMUL	c.c. 44-49	Multiplicity requirement. Real. Right justify. 1.E-01 (1.D-02) is recommended.
XSTART	c.c. 64-70	Magnitude at which to begin generating initial approximations. Real. Right justify. This is a special feature of the program and may be omitted.
XEND	c.c. 72-78	Magnitude at which to end the generating of initial approximations. Real. Right justify. This is a special feature of the program and may be omitted.
KCHECK	c.c. 80	This should be left blank.

TABLE XVII

CORRESPONDENCE OF NEWTON'S AND G.C.D. -
NEWTON'S INPUT DATA VARIABLES

Newton's MethodG.C.D. - Newton's Method

Coefficient Card

A(RA)

P(UP)

A(VA)

P(VP)

Initial Approximation Card

XZERO (RXZERO)

OIAPP (UOIAPP)

XZERO (VXZERO)

OIAPP (VOIAPP)

End Card

KCHECK

KCHECK

00000000	01111111	12222222	23333333	33344444	44444444	55555555	56666666	66666667	77777777	78
1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890
N O P O L Y	N P	N I A P	MAX	EP SG	EP SCNV	EP SQ	EP SMUL	X START	X END	K C H E C K
1	7	7	200	1.E-02	1.E-05	1.E-10	1.E-01	1.0E+01	2.0E+01	
1	7	7	200	1.D-03	1.D-10	1.D-20	1.D-02	1.0D+01	2.0D+01	

Figure 13. Control Card for G.C.D. - Newton's Method

3. Variables Used in G.C.D. - Newton's Method

The definitions of the major variables used in the G.C.D. - Newton's method are given in Table XVIII. For definitions of variables not listed in this table see the definitions of variables for the corresponding subroutine in Table VIII of Appendix B. The symbols and notation used here are the same as for Table VIII of Appendix B and are described in Appendix B, § 3.

4. Description of Program Output

The output from the G.C.D. - Newton's method is the same as for Newton's method as described in Appendix B, § 4 in addition to the following. The polynomial containing only simple roots; that is, after all multiple roots have been removed, is printed, coefficient of highest degree term first, under the heading "THE COEFFICIENTS OF $H(X) = P(X)/G.C.D. ARE$ " (Exhibit L). Also the test for zero in subroutine GCD is printed as read from the control card.

5. Informative and Error Messages

The output may contain informative messages printed as an aid to the user. These are:

"THE EPSILON (XXX) CHECK IN SUBROUTINE MULTIP INDICATES THAT ROOT(YY) = ZZZ IS NOT CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIPLICITY 0." XXX is the multiplicity requirement, YY is the number of the root obtained after the attempt to improve accuracy, and ZZZ is the value of the root after the attempt to improve accuracy. This message indicates that the value of the polynomial at the root

does not meet the requirement for multiplicities. The root, however, is usually a good approximation to the true root since convergence was obtained both before and after the attempt to improve accuracy.

"NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER XX." XX is the number of the polynomial. The message indicates that Newton's method did not converge to any root of this polynomial.

For a description of other messages see those of Newton's method given in Appendix B, § 5.

TABLE XVIII

VARIABLES USED IN THE G.C.D. - NEWTON'S METHOD

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
Main Program					
I01	I	I01	I		Unit number of input device
I02	I	I02	I		Unit number of output device
NOPOLY	I	NOPOLY	I		Number of the polynomial P(X)
NP	I	NP	I		Degree of P(X)
ANAME	A	ANAME	A		Name of method used (NEWTONS)
MAX	I	MAX	I		Maximum number of iterations permitted
NIAP	I	NIAP	I		Number of initial approximations to be read
EPSG	R	EPSG	R		Tolerance check for zero (0) in Subroutine GCD
KCHECK	I	KCHECK	I		Program control - KCHECK = 1 terminates execution
P	C	UP,VP	R		Array of coefficients of original polynomial, (P(X))
OIAPP	C	UOIAPP,VOIAPP	R		Array of initial approximations
NROOT	I	NROOT	I		Number of distinct roots found
IROOT	I	IROOT	I		Number of distinct roots found by the iterative process i.e. not as a result of Subroutine QUAD
EPSCNV	R	EPSCNV	R		Tolerance check for convergence
EPSQ	R	EPSQ	R		Tolerance check for zero (0) in QUAD
EPSMUL	R	EPSMUL	R		Tolerance check for multiplicities
XSTART	R	XSTART	R		Magnitude at which to begin generating approximations (initial)
XEND	R	XEND	R		Magnitude at which to end the generating of initial approximations
ROOT	C	UROOT,VROOT	R		Array of roots found
DP	C	UDP,VDP	R		Array of coefficients of derivative, P'(X), of P(X)
NDP	I	NDP	I		Degree of derivative P'(X)

TABLE XVII (Continued)

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition</u> <u>of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
RK	C	URK,VRK	R		Array of coefficients of the greatest common divisor of $P(X)$ and $P'(X)$
NRK	I	NRK	I		Degree of g.c.d. of $P(X)$ and $P'(X)$
H	C	UH,VH	R		Array of coefficients of quotient polynomial i.e. $P(X)/\text{g.c.d.}$
NH	I	NH	I		Degree of quotient polynomial $P(X)/\text{g.c.d.}$
RIAPP	C	URIAPP,VRIAPP	R		Array of approximations (initial or altered) producing convergence
DPOLY	C	UDPOLY,VDPOLY	R		Array of coefficients of deflated polynomial for which no roots were found
MULTI	I	MULTI	I		Array of multiplicities of each root
ND	I	ND	I		Program control and number of coefficients of deflated polynomial for which no zeros were found

Subroutine GCD

P	C	UP,VP	R	E	Array of coefficients of $P(X)$
NP	I	NP	I	E	Degree of $P(X)$
DP	C	UDP,VDP	R	E	Array of coefficients of $P'(X)$
NDP	I	NDP	I	E	Degree of $P'(X)$
R1	C	UR1,VR1	R		Array of coefficients of dividend polynomial
R2	C	UR2,VR2	R		Array of coefficients of divisor polynomial
N1	I	N1	I		Degree of dividend polynomial (R1)
N2	I	N2	I		Degree of divisor polynomial (R2)
T	C	UT,VT	R		Array of coefficients of difference polynomial $(R1 - U(R2))$
U	C	UU,VU	R		Quotient $(R1/R2)$
M	I	M	I		Degree of difference polynomial (T)

TABLE XVIII (Continued)

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
RK	C	URK,VRK	R	R	Array of coefficients of greatest common divisor of P(X) and P'(X)
NK	I	NK	I	R	Degree of g.c.d. of P(X) and P'(X)
EPSG	R	EPSG	R	E	Tolerance check for zero (0)
Subroutine DERIV					
A	C	UA,VA	R	E	Array of coefficients of polynomial, P(X)
NA	I	NA	I	E	Degree of P(X)
B	C	UB,VB	R	R	Array of coefficients of derivative, P'(X)
NB	I	NB	I	R	Degree of P'(X)
Subroutine MULTIP					
A	C	UA,VA	R	E	Array of coefficients of original polynomial, P(X)
NA	I	NA	I	E	Degree of original polynomial, P(X)
ROOT	C	UROOT,VROOT	R	E	Array of roots of P(X)
NR	I	NR	I	E	Number of roots (distinct) in array ROOT
EPSPX	R	EPSPX	R	E	Tolerance check for multiplicities
MULTI	I	MULTI	I	R	Array of multiplicities of each root
WORK	C	UWORK,VWORK	R		Working array for coefficients of current polynomial
NWORK	I	NWORK	I		Degree of polynomial whose coefficients are in WORK i.e. the current polynomial
B	C	UB,VB	R		Array of coefficients of newly deflated polynomial
C	C	UC,VC	R		Array containing sequence of values leading to the derivative
PX	C	UPX,VPX	R		Value of the polynomial at a point
DPX	C	UDPX,VDPX	R		Derivative of the polynomial at a point

TABLE XVIII (Continued)

<u>Single Precision</u>		<u>Double Precision</u>		<u>Disposition</u> <u>of Argument</u>	<u>Description</u>
<u>Variable</u>	<u>Type</u>	<u>Variable</u>	<u>Type</u>		
KANS	I	KANS	I		KANS = 1 implies convergence; KANS = 0 implies no convergence
I02	I	I02	I	C	Unit number of output device
Subroutine DIVIDE					
F	C	UF,VF	R	E	Array of coefficients of dividend polynomial
NF	I	NF	I	E	Degree of dividend polynomial
G	C	UG,VG	R	E	Array of coefficients of divisor polynomial
NG	I	NG	I	E	Degree of divisor polynomial
H	C	UH,VH	R	R	Array of coefficients of quotient polynomial
NH	I	NH	I	R	Degree of quotient polynomial
TERM	C	UTERM,VTERM	R		Dummy variable used for temporary storage
NNF	I	NNF	I		Degree of dividend polynomial to be deflated
X	C	UX,VX	R		Value at which to deflate polynomial F(X)
C	C	UC,VC	R		Array of sequence of values leading to the derivative
PX	C	UPX,VPX	R		Value of polynomial at the point X
DPX	C	UDPX,VDPX	R		Derivative of the polynomial at the point X

MAIN PROGRAM

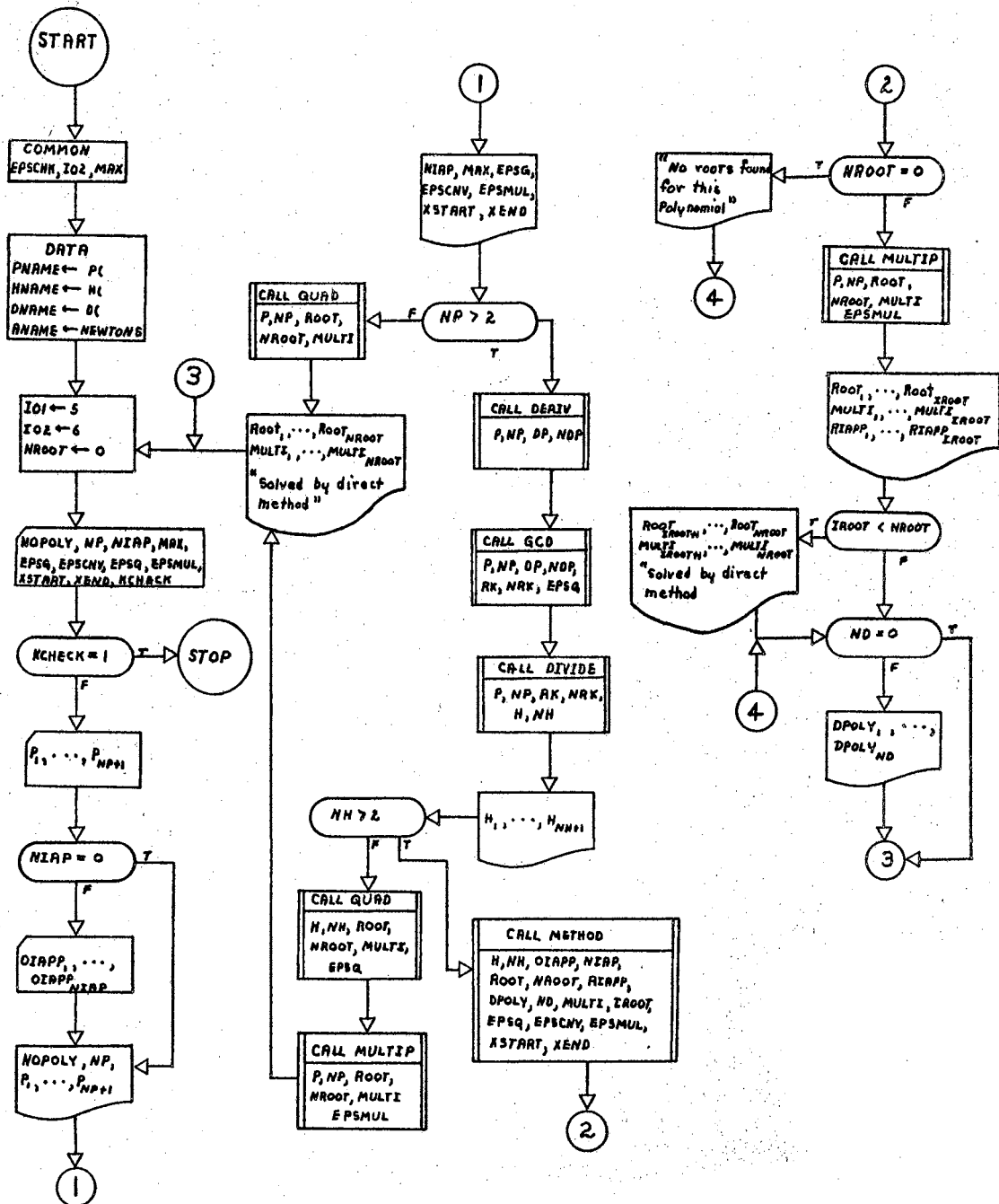


Figure 13.1. Flow Charts for G.C.D. - Newton's Method

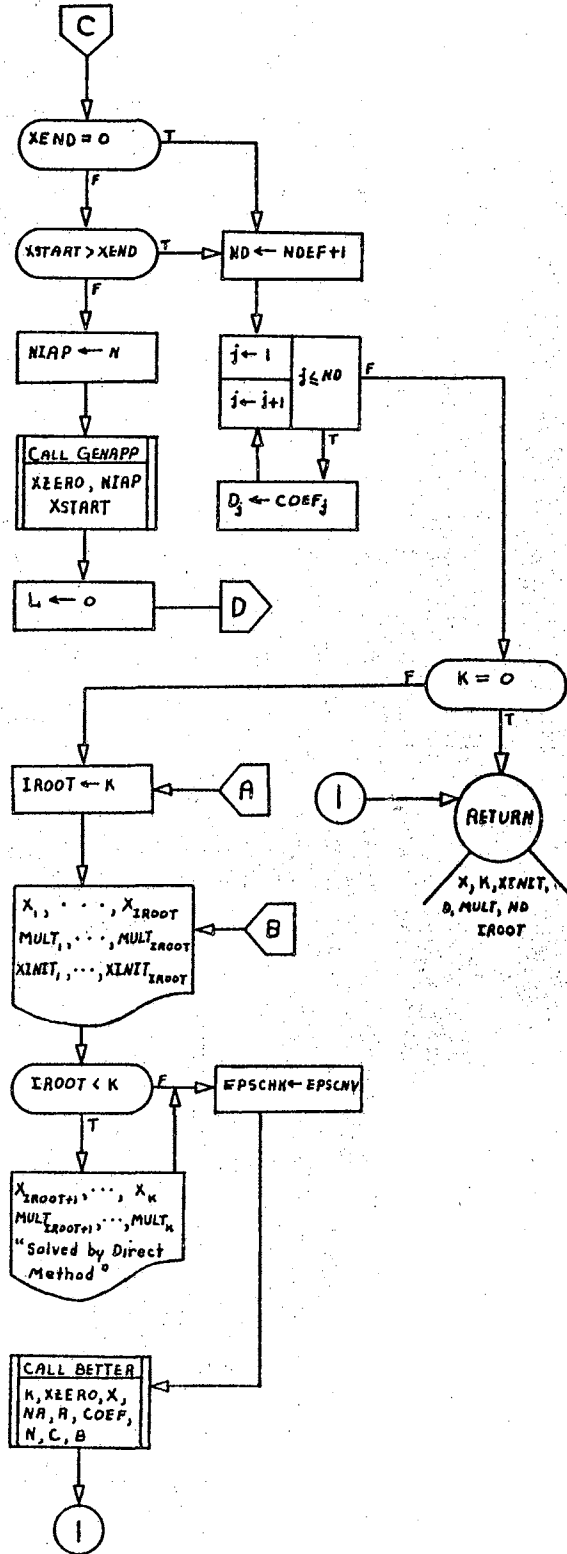
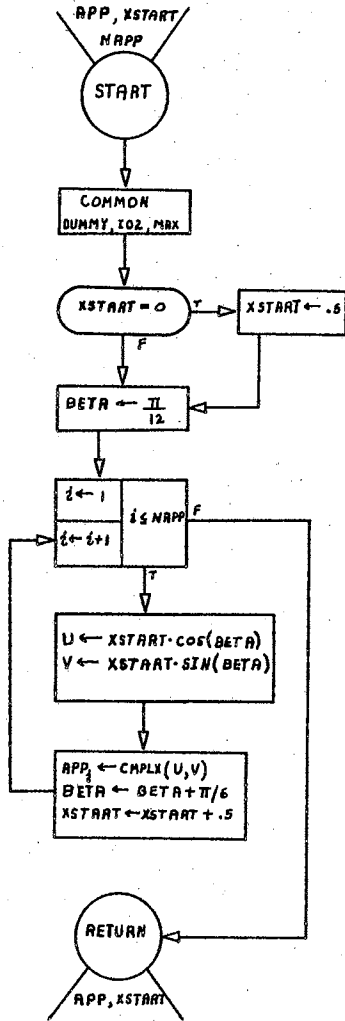


Figure 13.1. (Continued)

GENAPP



ALTER

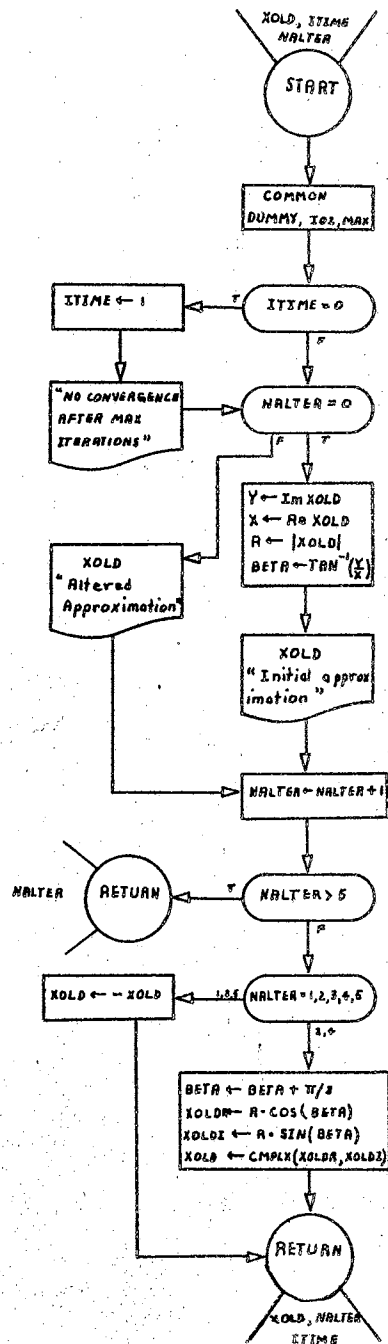


Figure 13.1. (Continued)

BETTER

CHECK

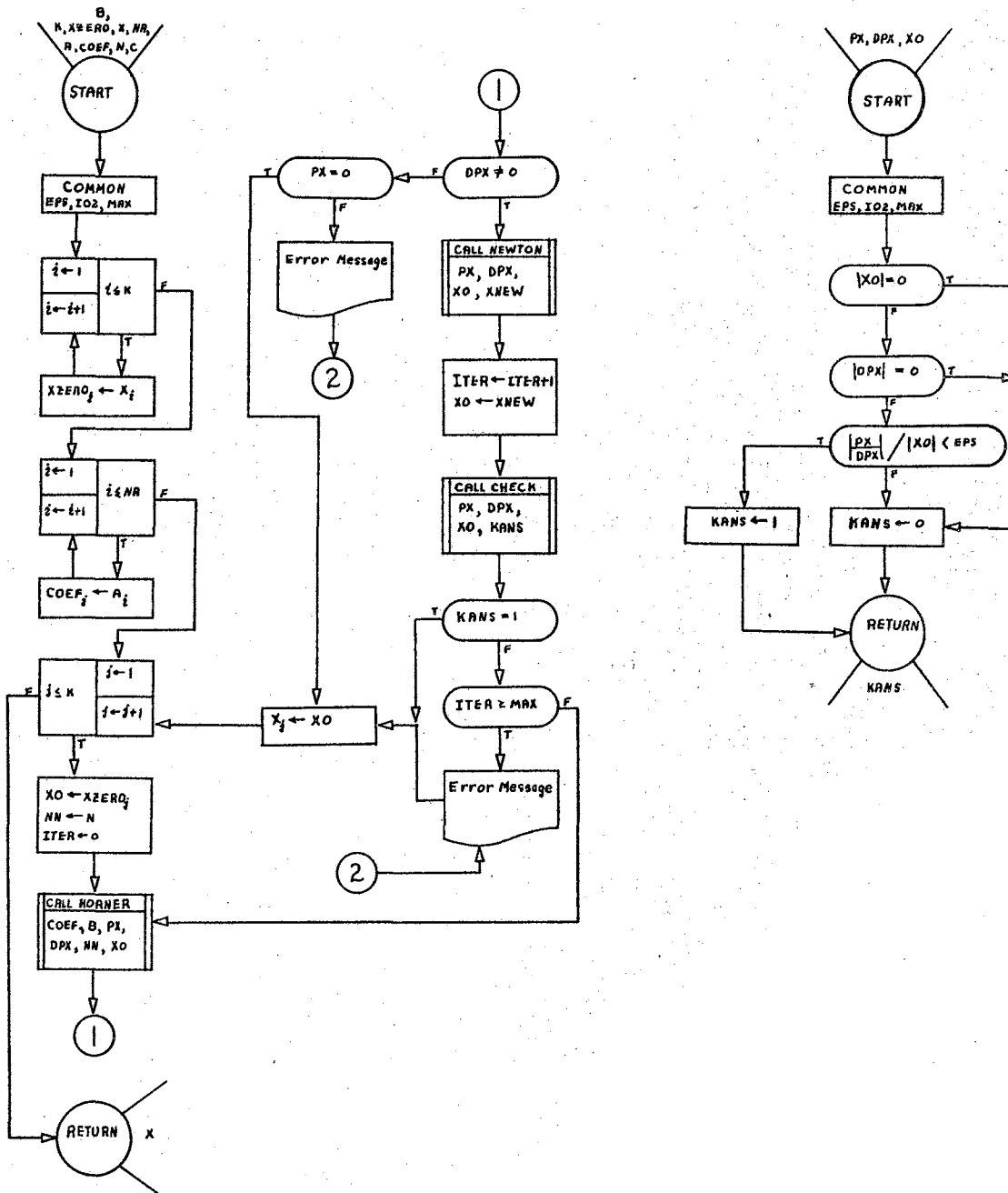
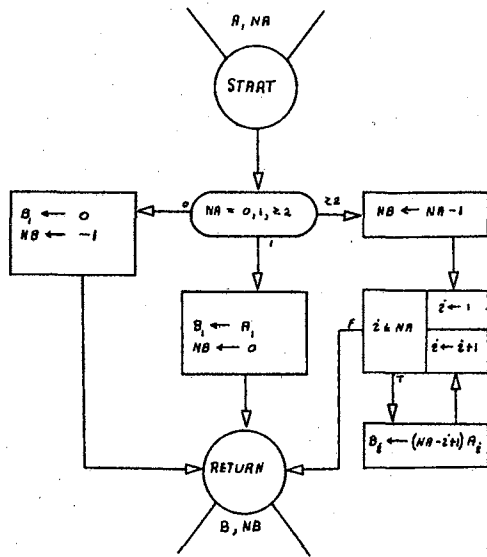


Figure 13.1. (Continued)

DERIV



MULTIP

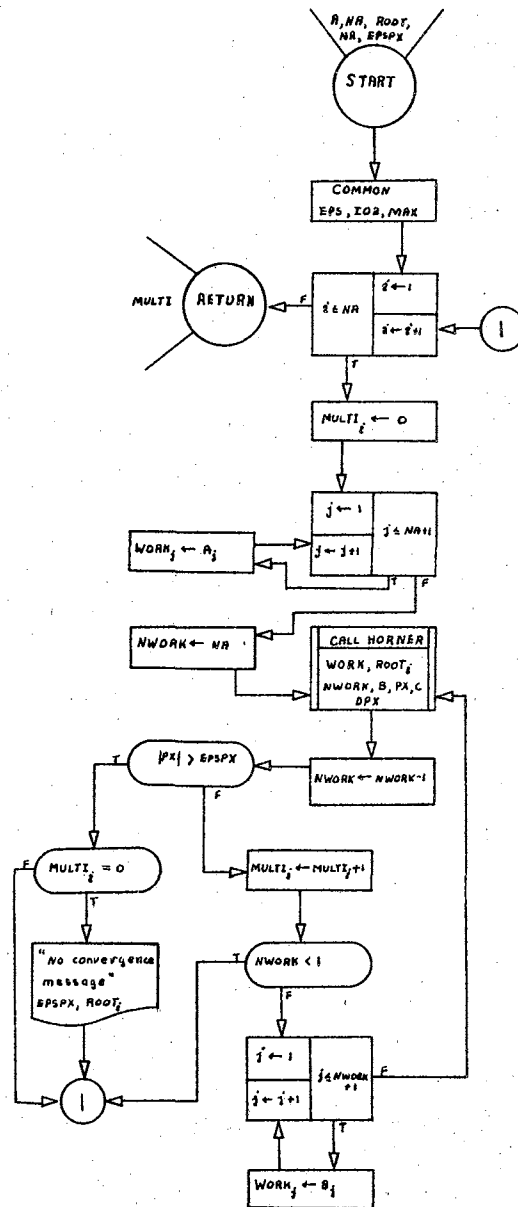


Figure 13.1. (Continued)

GCD

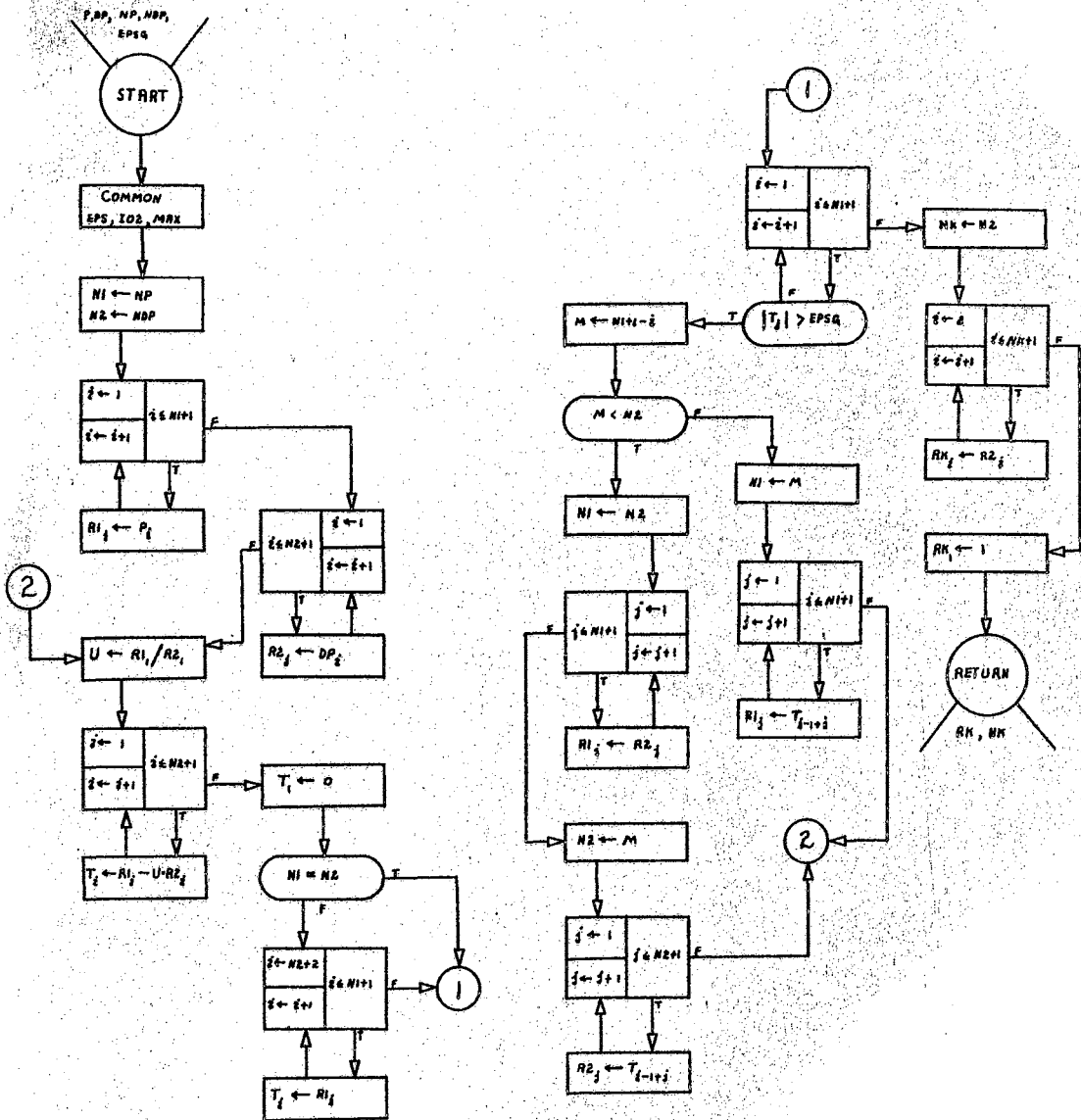


Figure 13.1. (Continued)

DIVIDE

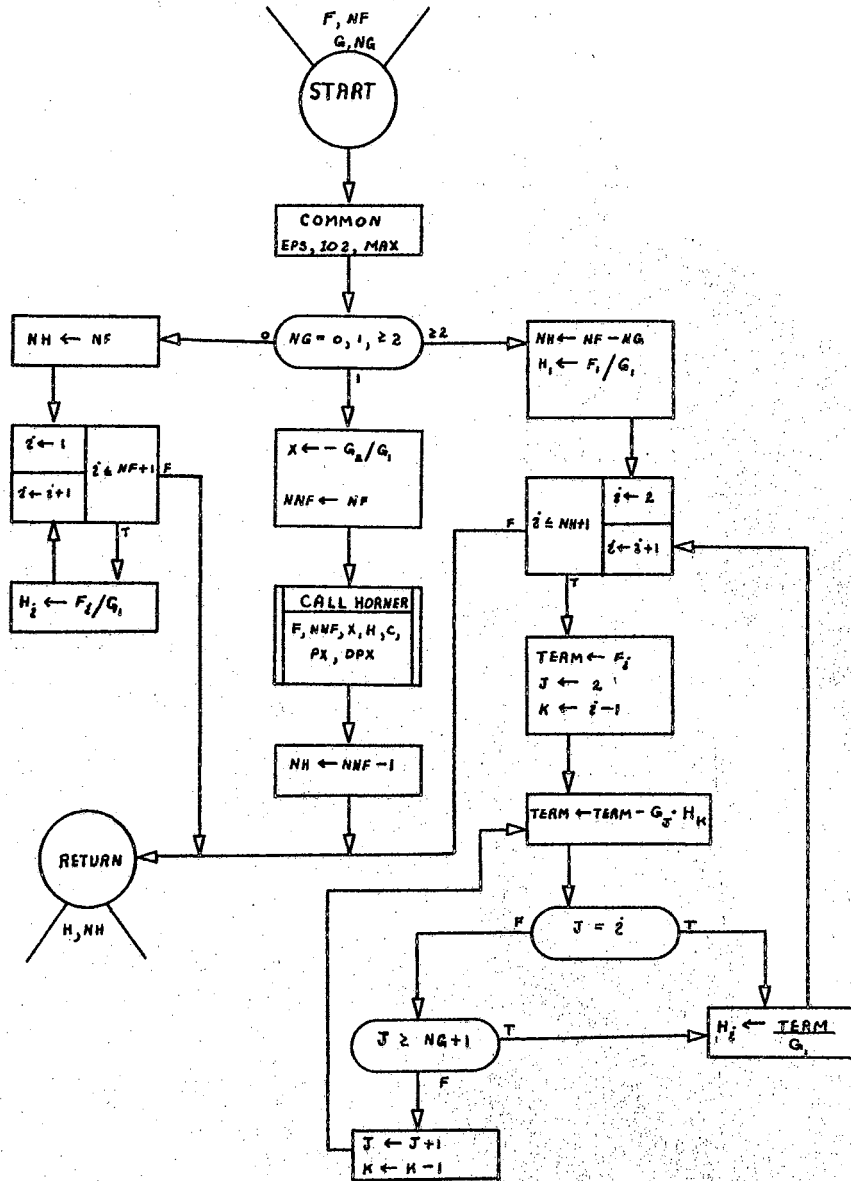
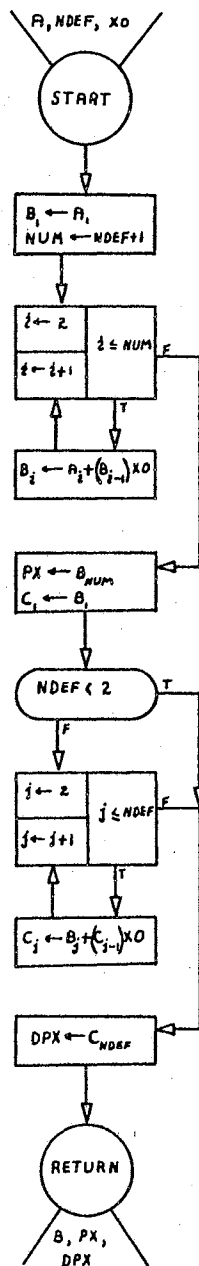


Figure 13.1. (Continued)

HORNER



NEWTON

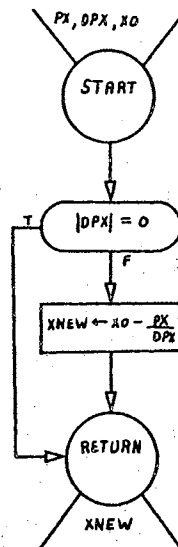
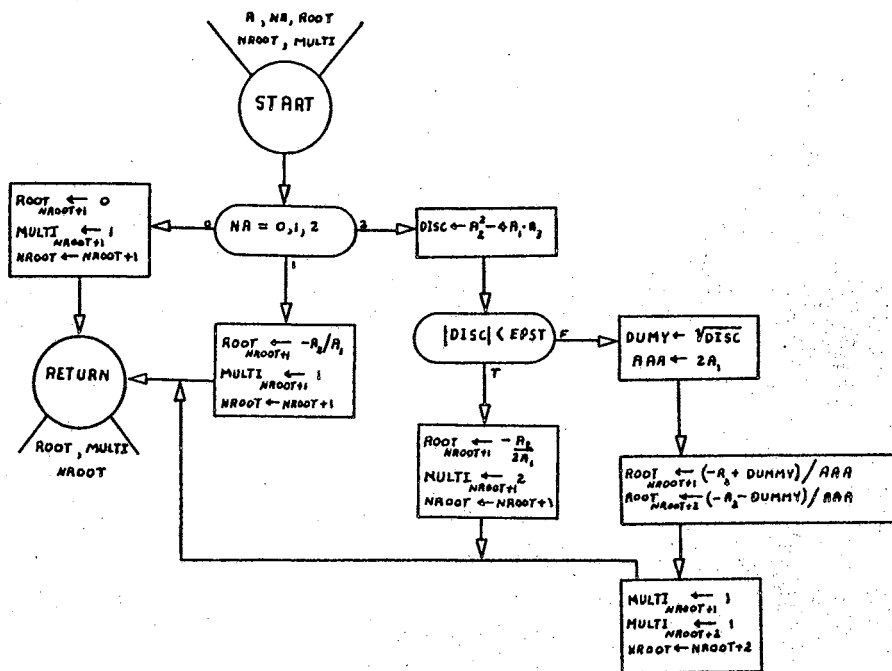


Figure 13.1. (Continued)

QUAD



COMSQT

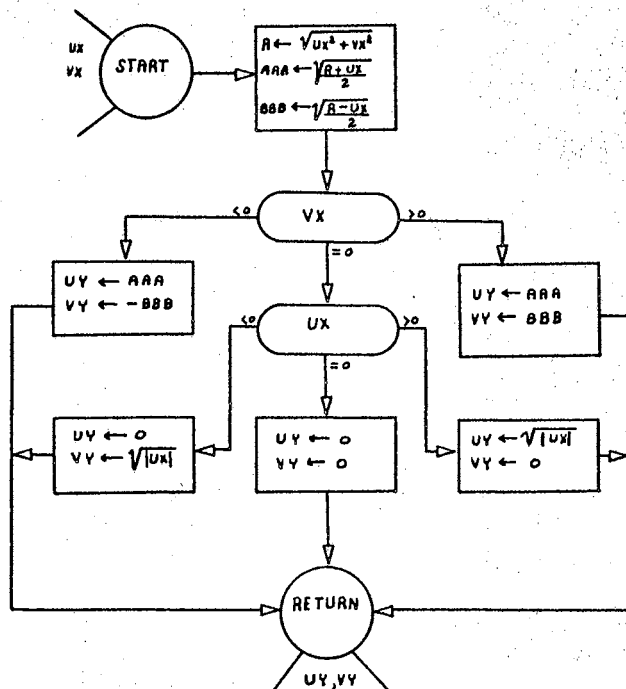


Figure 13.1. (Continued)

TABLE XVIII-A

SINGLE PRECISION PROGRAM FOR G.C.D. - NEWTON'S METHOD

\$JOB 10414

```

C *****
C *
C * SINGLE PRECISION PROGRAM FOR G.C.D. - NEWTON'S METHOD
C *
C *
C * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
C * POLYNOMIAL OF MAXIMUM DEGREE 25. ALL MULTIPLE ROOTS ARE REMOVED BY
C * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL *
C * AND ITS DERIVATIVE. THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED *
C * AND THEIR MULTIPLICITIES DETERMINED.
C *
C *****
1  COMPLEX P,OIAPP,ROOT,DP,RK,RIAPP,H,APPNO,DPOLY
2  DIMENSION P(26),OIAPP(25),ROOT(25),MULTI(25),DP(26),RK(26),RIAPP(2
   15),H(26),DPOLY(26),ANAME(2)
3  COMMON EPSCCHK,I02,MAX
4  DATA PNAME,HNAME,DNAME/2HP(,2HH(,2HD(/
5  DATA ANAME(1),ANAME(2)/4HNEWT,4HONS /
6  I01=5
7  I02=6
8  10 NROOT=0
9  READ(I01,1000) NOPOLY,NP,NIAP,MAX,EPSPG,EPSCNV,EPSPQ,EPSPUL,XSTART,X
   IEND,KCHECK
10 IF(KCHECK.EQ.1) STOP
11 KKK=NP+1
12 READ(I01,1010) (P(I),I=1,KKK)
13 IF(NIAP.EQ.0) GO TO 20
14 READ(I01,1020) (OIAPP(I),I=1,NIAP)
15 20 KKK=NP+1
16 WRITE(I02,1030) ANAME(1),ANAME(2),NOPOLY,NP
17 WRITE(I02,1035) (PNAME,I,P(I),I=1,KKK)
18 WRITE(I02,2060)
19 WRITE(I02,2000) NIAP
20 WRITE(I02,2010) MAX
21 WRITE(I02,2070) EPSPG
22 WRITE(I02,2020) EPSCNV
23 WRITE(I02,2030) EPSPUL
24 WRITE(I02,2040) XSTART
25 WRITE(I02,2050) XEND
26 IF(NP.GT.2) GO TO 90
27 CALL QUAD(P,NP,ROOT,NROOT,MULTI,EPSPQ)
28 85 WRITE(I02,1037)
29 WRITE(I02,1086) (I,ROOT(I),MULTI(I),I=1,NROOT)
30 GO TO 10
31 90 CALL DERIV(P,NP,DP,NDP)
32 CALL GCD(P,NP,DP,NDP,RK,NRK,EPSPG)
33 CALL DIVIDE(P,NP,RK,NRK,H,NH)
34 KKK=NH+1
35 WRITE(I02,1060)
36 WRITE(I02,1035) (HNAME,I,H(I),I=1,KKK)
37 IF(NH.GT.2) GO TO 150
38 CALL QUAD(H,NH,ROOT,NROOT,MULTI,EPSPQ)
39 CALL MULTIP(P,NP,ROOT,NROOT,MULTI,EPSPUL)
40 GO TO 85
41 150 CALL METHOD(H,NH,OIAPP,NIAP,ROOT,NROOT,RIAPP,DPOLY,MULTI,ND,IROOT,
   IEPSPQ,XSTART,XEND,EPSCNV,EPSPUL)
42 IF(NROOT.NE.0) GO TO 170
43 WRITE(I02,1070) NOPOLY

```

TABLE XVIII-A (Continued)

```

44      GO TO 200
45      170 CALL MULTIP(P,NP,ROOT,NROOT,MULTI,EPSTMUL)
46          WRITE(IO2,1065)
47          WRITE(IO2,1080)
48          WRITE(IO2,1085) (I,ROOT(I),MULTI(I),RIAPP(I),I=1,IROOT)
49          KKK=IROOT+1
50          IF(IROOT.LT.NROOT) WRITE(IO2,1086) (I,ROOT(I),MULTI(I),I=KKK,NROOT
51      200 IF(ND.EQ.0) GO TO 10
52      230 WRITE(IO2,1090)
53          WRITE(IO2,1035) (DNAME,J,DPOLY{J},J=1,ND)
54          GO TO 10
55      1080 FORMAT(///1X,13HROOTS OF P(X),37X,14HMULTIPLICITIES,11X,21HINITIAL
56      1085 FORMAT(2X,5HROOT(I,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,10X,E14.
57      1086 FORMAT(2X,5HROOT(I,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,11X,23HS
58      1000 FORMAT(3(I2,1X),9X,I3,1X,4(E6.0,1X),13X,2(E7.0,1X),I1)
59      1010 FORMAT(2E30.0)
60      1020 FORMAT(2E30.0)
61      1030 FORMAT(1H1,10X,41HGREATEST COMMON DIVISOR METHOD USED WITH ,2(A4),
62      1035 FORMAT(3X,A2,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
63      2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,I2)
64      2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
65      2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,E9.2)
66      2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,E9.2)
67      2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,E9.2)
68      2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,E9.2)
69      2060 FORMAT(//1X)
70      2070 FORMAT(1X,31HTEST FOR ZERO IN SUBROUTINE GCD,3X,E9.2)
71      1037 FORMAT(///,1X,13HZEROS OF P(X),37X,14HMULTIPLICITIES//)
72      1060 FORMAT(///1X,42HTHE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE//)
73      1070 FORMAT(///1X,42HNO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
74      1065 FORMAT(///1X,61HAFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
75      1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
76      END

```


TABLE XVIII-A (Continued)

```

77      SUBROUTINE METHOD(A,N,XZERO,NIAP,X,K,XINIT,D,MULT,ND,IROOT,EPSQ,XS
      1TART,XEND,EPSCNV,EPSMUL)
C      *****
C      *
C      * THIS SUBROUTINE USES NEWTON'S METHOD TO EXTRACT THE ZEROS OF A
C      * POLYNOMIAL. A SEQUENCE OF APPROXIMATIONS IS CONSTRUCTED CONVERGING TO A
C      * ZERO OF THE POLYNOMIAL BY USING THE ITERATION FORMULA
C      *
C      *          X(N+1) = X(N)-P(X(N))/P'(X(N)).
C      *
C      *****
78      COMPLEX A,XZERO,B,COEF,X,XINIT,C,D,PX,DPX,XNEW,XO
79      DIMENSION A(26),B(26),C(26),D(26),COEF(26),MULT(25),XZERO(25),X(25
      1),XINIT(25)
80      COMMON EPSCHK,IO2,MAX
81      IF(NIAP.NE.0) GO TO 1
82      NIAP=N
83      CALL GENAPP(XZERO,NIAP,XSTART)
84      1 NA=N+1
85      NDEF=N
86      L=1
87      ITER=0
88      NROOT=0
89      IROOT=0
90      IALTER=0
91      ITIME=0
92      ND=0
93      K=0
94      XO=XZERO(L)
95      DO 5 I=1,NA
96      5 COEF(I)=A(I)
97      10 CALL HORNER(COEF,B,PX,DPX,NDEF,XO,C)
98      ABPX=CABS(PX)
99      ABDPX=CABS(DPX)
100     IF(ABDPX.NE.0.0) GO TO 20
101     IF(ABPX.EQ.0.0) GO TO 70
102     GO TO 110
103     20 CALL NEWTON(PX,DPX,XO,XNEW)
104     ITER=ITER+1
105     XO=XNEW
106     EPSCHK=EPSCNV
107     CALL CHECK(PX,DPX,XO,KANS)
108     IF(KANS.EQ.1) GO TO 70
109     IF(ITER.GE.MAX) GO TO 40
110     GO TO 10
111     40 CALL ALTER(XZERO(L),IALTER,ITIME)
112     IF(IALTER.GT.5) GO TO 110
113     XO=XZERO(L)
114     ITER=0
115     GO TO 10
116     60 ND=NDEF+1
117     DO 65 J=1,ND
118     65 D(J)=COEF(J)
119     GO TO 140
120     70 NROOT=NROOT+1
121     K=K+1
122     MULT(K)=1
123     X(K)=XO
124     XINIT(K)=XZERO(L)
125     CALL HORNER(COEF,B,PX,DPX,NDEF,XO,C)
126     80 IF(NROOT.GE.N) GO TO 147

```

TABLE XVIII-A (Continued)

127	NDEF=NDEF-1	
128	NUM=NDEF+1	
129	DO 105 I=1,NUM	294
130	105 COEF(I)=B(I)	296
131	CALL HORNER(COEF,B,PX,DPX,NDEF,XO,C)	
132	ABPX=CABS(PX)	
133	ABDPX=CABS(DPX)	
134	IF(ABDPX.NE.0.0) GO TO 107	
135	IF(ABPX.EQ.0.0) GO TO 130	
136	GO TO 110	
137	107 CONTINUE	
138	EPSCHK=EPSMUL	
139	CALL CHECK(PX,DPX,XO,KANS)	
140	IF(KANS.EQ.1) GO TO 130	300
141	110 IF(NDEF.GT.2) GO TO 113	
142	IROOT=K	
143	CALL QUAD(COEF,NDEF,X,K,MULT,EPSQ)	
144	GO TO 150	
145	113 IF(L.LT.NIAP) GO TO 115	
146	IF(XEND.EQ.0.0) GO TO 60	
147	IF(XSTART.GT.XEND) GO TO 60	
148	NIAP=N	
149	CALL GENAPP(XZERO,NIAP,XSTART)	
150	L=0	
151	115 L=L+1	
152	XO=XZERO(L)	312
153	ITER=0	316
154	IALTER=0	320
155	GO TO 10	324
156	130 MULT(K)=MULT(K)+1	328
157	NROOT=NROOT+1	332
158	GO TO 80	336
159	140 IF(K.EQ.0) GO TO 160	338
160	147 IROOT=K	
161	150 WRITE(I02,I025)	
162	WRITE(I02,I050)	380
163	WRITE(I02,I060) (I,X(I),MULT(I),XINIT(I),I=1,IROOT)	
164	KKK=IROOT+1	
165	IF(IROOT.LT.K) WRITE(I02,I062) (I,X(I),MULT(I),I=KKK,K)	
166	EPSCHK=EPSCNV	
167	CALL BETTER(K,XZERO,X,NA,A,COEF,N,C,B)	
168	160 RETURN	
169	1025 FORMAT(///1X,61HBEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS 10F P(X) ARE)	
170	1050 FORMAT(///1X,13HZEROS OF P(X),38X,14HMULTIPLICITIES,11X,21HINITIAL 1 APPROXIMATION/)	
171	1060 FORMAT(3X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,9X,I2,12X,E14.7 1,3H + ,E14.7,2H I)	
172	1062 FORMAT(3X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,9X,I2,12X,23HSD 1LVED BY DIRECT METHOD)	
173	END	450

TABLE XVIII-A (Continued)

```

174      SUBROUTINE GENAPP(APP,NAPP,XSTART)
C      *****
C      *
C      * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE *
C      * DEGREE OF THE ORIGINAL POLYNOMIAL. *
C      *
C      *****
175      COMPLEX APP
176      COMPLEX CMLX
177      DIMENSION APP(25)
178      COMMON DUMMY,IO2,MAX
179      IF(XSTART.EQ.0.0) XSTART=0.5
180      BETA=0.2617994
181      DO 10 I=1,NAPP
182      U=XSTART*COS(BETA)
183      V=XSTART*SIN(BETA)
184      APP(I)=CMLX(U,V)
185      BETA=BETA+0.5235988
186      10 XSTART=XSTART+0.5
187      RETURN
188      END

189      SUBROUTINE ALTER(XOLD,NALTER,ITIME)
C      *****
C      *
C      * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO *
C      * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
C      *
C      *****
190      COMPLEX XOLD
191      COMPLEX CMLX
192      COMMON DUMMY,IO2,MAX
193      IF(ITIME.NE.0) GO TO 5
194      ITIME=1
195      WRITE(IO2,1010) MAX
196      5 IF(NALTER.EQ.0) GO TO 10
197      WRITE(IO2,1000) XOLD
198      GO TO 20
199      10 Y=AIMAG(XOLD)
200      X=REAL(XOLD)
201      R=CABS(XOLD)
202      BETA=ATAN2(Y,X)
203      WRITE(IO2,1020) XOLD
204      20 NALTER=NALTER+1
205      IF(NALTER.GT.5) RETURN
206      GO TO (30,40,30,40,30),NALTER
207      30 XOLD=-XOLD
208      GO TO 50
209      40 BETA=BETA+1.0471976
210      XOLDR=R*COS(BETA)
211      XOLDI=R*SIN(BETA)
212      XOLD=CMLX(XOLDR,XOLDI)
213      50 RETURN
214      1000 FORMAT(1X,E14.7,3H + ,E14.7,2H I,10X,21HALTERED APPROXIMATION)
215      1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
1TER ,13,12H ITERATIONS.//)
216      1020 FORMAT(1X,E14.7,3H + ,E14.7,2H I,10X,21HINITIAL APPROXIMATION)
217      END

```


TABLE XVIII-A (Continued)

```

241      SUBROUTINE CHECK(PX,DPX,XO,KANS)
C      *****
C      *
C      * THIS SUBROUTINE CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
C      * IMATIONS BY TESTING THE EXPRESSION
C      * ABSOLUTE VALUE OF (P(X(N))/P'(X(N)))/ABSOLUTE VALUE OF X(N+1).
C      * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
C      *
C      *****
242      COMPLEX PX,DPX,XO
243      COMMON EPS,IO2,MAX
244      IF(CABS(XO).EQ.0.) GO TO 25
245      DDD=CABS(DPX)
246      IF(DDD.EQ.0.0) GO TO 25
247      IF (CABS(PX/DPX)/CABS(XO).LT.EPS) GO TO 10
248      KANS=0
249      RETURN
250      10 KANS=1
251      RETURN
252      25 KANS=0
253      RETURN
254      END

```

748

760

764

768

772

780

TABLE XVIII-A (Continued)

```

255      SUBROUTINE BETTER(K,XZERO,X,NA,A,COEF,N,C,B)
C      *****
C      *
C      * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND *
C      * BY USING THEM AS INITIAL APPROXIMATIONS WITH NEWTON'S METHOD APPLIED TO *
C      * THE FULL, UNDEFLATED POLYNOMIAL. *
C      *
C      *****
256      COMPLEX XZERO,X,A,COEF,C,B,XO,PX,DPX,XNEW      804
257      DIMENSION XZERO(25),X(25),A(26),COEF(26),C(26),B(26)
258      COMMON EPS,I02,MAX
259      DO 10 I=1,K      812
260      10 XZERO(I)=X(I)      815
261      DO 20 I=1,NA      820
262      20 COEF(I)=A(I)      824
263      DO 50 J=1,K      828
264      XO=XZERO(J)      832
265      NN=N      834
266      ITER=0      836
267      30 CALL HORNER(COEF,B,PX,DPX,NN,XO,C)
268      ABPX=CABS(PX)
269      ABDPX=CABS(DPX)
270      IF(ABDPX.NE.0.0) GO TO 33
271      IF(ABPX.EQ.0.0) GO TO 40
272      GO TO 34
273      33 CALL NEWTON(PX,DPX,XO,XNEW)
274      ITER=ITER+1      856
275      XO=XNEW      860
276      CALL CHECK(PX,DPX,XO,KANS)
277      IF(KANS.EQ.1) GO TO 40      844
278      IF(ITER.GE.MAX) GO TO 35
279      GO TO 30      864
280      34 WRITE(I02,1112) XO
281      35 WRITE(I02,100) J,XZERO(J)
282      WRITE(I02,200) MAX
283      40 X(J)=XO      868
284      50 CONTINUE      872
285      RETURN      876
286      1112 FORMAT(1H0,36H THE VALUE OF THE DERIVATIVE AT XO = ,E14.7,3H + ,E14
1.7,2H I,10H IS ZERO.)
287      100 FORMAT(42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,E14
1.7,3H + ,E14.7,2H I,18H DID NOT CONVERGE.)
288      200 FORMAT(33H THE PRESENT APPROXIMATION AFTER ,I3,29H ITERATIONS IS P
PRINTED BELOW.)
289      END      880

```

TABLE XVIII-A (Continued)

```

290      SUBROUTINE DERIV(A,NA,B,NB)
C      *****
C      *
C      * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF *
C      * ITS DERIVATIVE P'(X). *
C      *
C      *****
291      COMPLEX A,B
292      DIMENSION A(26),B(26)
293      IF(NA.GE.2) GO TO 30
294      IF(NA.EQ.1) GO TO 30
295      B(1)=0.0
296      NB=-1
297      GO TO 50
298      20 B(1)=A(1)
299      NB=0
300      GO TO 50
301      30 NB=NA-1
302      DO 40 I=1,NA
303      BBB=NA-I+1
304      40 B(I)=BBB*A(I)
305      50 RETURN
306      END

307      SUBROUTINE DIVIDE(F,NF,G,NG,H,NH)
C      *****
C      *
C      * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE *
C      * QUOTIENT POLYNOMIAL H(X) = F(X)/G(X). *
C      *
C      *****
308      COMPLEX F,G,H,X,C,PX,DPX,TERM
309      DIMENSION F(26),G(26),H(26),C(26)
310      COMMON EPS,I02,MAX
311      IF(NG.GE.2) GO TO 60
312      IF(NG.EQ.1) GO TO 30
313      NH=NF
314      KKK=NF+1
315      DO 20 I=1,KKK
316      20 H(I)=F(I)/G(1)
317      GO TO 100
318      30 X=-G(2)/G(1)
319      NNF=NF
320      CALL HORNER(F,H,PX,DPX,NNF,X,C)
321      NH=NNF-1
322      GO TO 100
323      60 NH=NF-NG
324      H(1)=F(1)/G(1)
325      KKK=NH+1
326      DO 90 I=2,KKK
327      TERM=F(I)
328      J=2
329      K=I-1
330      70 TERM=TERM-G(J)*H(K)
331      IF(J.EQ.1) GO TO 90
332      IF(J.GE.NG+1) GO TO 90
333      J=J+1
334      K=K-1
335      GO TO 70
336      90 H(I)=TERM/G(1)
337      100 RETURN
338      END

```

TABLE XVIII-A (Continued)

```

339 SUBROUTINE QUAD(A,NA,ROOT,NROOT,MULTI,EPST)
C *****
C *
C * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
C * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE *
C * QUADRATIC IS DONE USING THE QUADRATIC FORMULA. *
C *
C *****
340 COMPLEX A,DISC,ROOT,DUMMY,AAA
341 COMPLEX CSQRT
342 DIMENSION A(26),ROOT(25),MULTI(25)
343 IF(NA.EQ.2) GO TO 7
344 IF(NA.EQ.1) GO TO 5
345 ROOT(NROOT+1)=0.0
346 MULTI(NROOT+1)=1
347 NROOT=NROOT+1
348 GO TO 50
349 5 ROOT(NROOT+1)=-A(2)/A(1)
350 MULTI(NROOT+1)=1
351 NROOT=NROOT+1
352 GO TO 50
353 7 DISC=A(2)*A(2)-(4.0*A(1)*A(3))
354 BBB=CABS(DISC)
355 IF(BBB.LT.EPST) GO TO 10
356 DUMMY=CSQRT(DISC)
357 AAA=2.0*A(1)
358 ROOT(NROOT+1)=(-A(2)+DUMMY)/AAA
359 ROOT(NROOT+2)=(-A(2)-DUMMY)/AAA
360 MULTI(NROOT+1)=1
361 MULTI(NROOT+2)=1
362 NROOT=NROOT+2
363 GO TO 50
364 10 ROOT(NROOT+1)=(-A(2))/(2.0*A(1))
365 MULTI(NROOT+1)=2
366 NROOT=NROOT+1
367 50 RETURN
368 END

```


TABLE XVIII-A (Continued)

```

369      SUBROUTINE GCD(P,NP,DP,NDP,RK,NK,EPG)
C      *****
C      *
C      * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG.
C      * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND
C      * DP(X).
C      *
C      *****
370      COMPLEX R1,R2,RK,U,T,P,DP
371      DIMENSION R1(26),R2(26),T(26),RK(26),P(26),DP(26)
372      COMMON EPS,IO2,MAX
373      N1=NP
374      N2=NDP
375      KKK=N1+1
376      DO 10 I=1,KKK
377      10 R1(I)=P(I)
378      KKK=N2+1
379      DO 20 I=1,KKK
380      20 R2(I)=DP(I)
381      50 U=R1(1)/R2(1)
382      KKK=N2+1
383      DO 70 I=1,KKK
384      70 T(I)=R1(I)-U*R2(I)
385      T(1)=0.0
386      IF(N1.EQ.N2) GO TO 90
387      KKK=N1+1
388      NNN=N2+2
389      DO 80 I=NNN,KKK
390      80 T(I)=R1(I)
391      90 KKK=N1+1
392      DO 100 I=1,KKK
393      BBB=CABS(T(I))
394      IF(BBB.GT.EPSG) GO TO 120
395      100 CONTINUE
396      NK=N2
397      KKK=NK+1
398      DO 110 I=2,KKK
399      110 RK(I)=R2(I)/R2(1)
400      RK(1)=1.0
401      GO TO 200
402      120 M=N1+1-I
403      IF(M.LT.N2) GO TO 160
404      N1=M
405      KKK=N1+1
406      NNN=I-1
407      DO 150 J=1,KKK
408      150 R1(J)=T(NNN+J)
409      GO TO 50
410      160 N1=N2
411      KKK=N1+1
412      DO 170 J=1,KKK
413      170 R1(J)=R2(J)
414      N2=M
415      KKK=N2+1
416      NNN=I-1
417      DO 180 J=1,KKK
418      180 R2(J)=T(NNN+J)
419      GO TO 50
420      200 RETURN
421      END

```

TABLE XVIII-A (Continued)

```

422 SUBROUTINE MULTIP(A,NA,ROOT,NR,MULTI,EPSPX)
C *****
C *
C * GIVEN NR ZEROS OF A POLYNOMIAL, SUBROUTINE MULTIP COMPUTES THEIR *
C * MULTIPLICITIES. *
C *
C *****
423 COMPLEX A,ROOT,WORK,C,PX,DPX,B
424 DIMENSION A(26),ROOT(25),WORK(26),MULTI(25),B(26),C(26)
425 COMMON EPS,IQ2,MAX
426 DO 50 I=1,NR
427 MULTI(I)=0
428 KKK=NA+1
429 DO 20 J=1,KKK
430 20 WORK(J)=A(J)
431 NWORK=NA
432 25 CALL HORNER(WORK,B,PX,DPX,NWORK,ROOT(I),C)
433 NWORK=NWORK-1
434 BBB=CABS(PX)
435 IF(BBB.GT.EPSPX) GO TO 45
436 MULTI(I)=MULTI(I)+1
437 IF(NWORK.LT.1) GO TO 50
438 KKK=NWORK+1
439 DO 40 J=1,KKK
440 40 WORK(J)=B(J)
441 GO TO 25
442 45 IF(MULTI(I).EQ.0) WRITE(IQ2,1000) EPSPX,I,ROOT(I)
443 50 CONTINUE
444 100 RETURN
445 1000 FORMAT(///14H THE EPSILON (,E14.7,49H) CHECK IN SUBROUTINE MULTIP
1 INDICATES THAT ROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I/80H IS NOT C
2 LOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIP LIC
3 ITY 0//)
446 END

```

SENTRY

TABLE XVIII-B

DOUBLE PRECISION PROGRAM FOR G.C.D. - NEWTON'S METHOD

\$JOB 10414

```

C *****
C *
C * DOUBLE PRECISION PROGRAM FOR G.C.D. - NEWTON'S METHOD
C *
C *
C * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
C * POLYNOMIAL OF MAXIMUM DEGREE 25. ALL MULTIPLE ROOTS ARE REMOVED BY
C * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL
C * AND ITS DERIVATIVE. THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED
C * AND THEIR MULTIPLICITIES DETERMINED.
C *
C *****
1  DOUBLE PRECISION UP,VP,UOIAPP,VOIAPP,UROOT,VROOT,UDP,VDP,URK,VRK,U
   IRIAPP,VRIAPP,UH,VH,UDPOLY,VDPOLY,EPSCHK,UDUMMY,VDUMMY,EPSQ,EPSMUL,
   2  2EPSG,EPSCNV,XSTART,XEND
   DIMENSION ANAME(2),UP(26),VP(26),UOIAPP(25),VOIAPP(25),UROOT(25),V
   3  IROOT(25),UDP(26),VDP(26),URK(26),VRK(26),URIAPP(25),VRIAPP(25),UH(
   4  226),VH(26),UDPOLY(26),VDPOLY(26),MULTI(25)
   COMMON EPSCHK,IO2,MAX
   5  DATA PNAME,HNAME,DNAME/2HP(,2HH(,2HD(/
   6  DATA ANAME(1),ANAME(2)/4HNEWT,4HONS /
   7  IO1=5
   8  IO2=6
   9  10 NRROOT=0
   READ(IO1,1000) NOPOLY,NP,NIAP,MAX,EPSP,EPSCNV,EPSP,EPSPMUL,XSTART,X
   10  IEND,KCHECK
   IF(KCHECK.EQ.1) STOP
   11  KKK=NP+1
   12  READ(IO1,1010) (UP(I),VP(I),I=1,KKK)
   13  IF(NIAP.EQ.0) GO TO 20
   14  READ(IO1,1020) (UOIAPP(I),VOIAPP(I),I=1,NIAP)
   15  20 KKK=NP+1
   16  WRITE(IO2,1030) ANAME(1),ANAME(2),NOPOLY,NP
   17  WRITE(IO2,1035) (PNAME,I,UP(I),VP(I),I=1,KKK)
   18  WRITE(IO2,2060)
   19  WRITE(IO2,2000) NIAP
   20  WRITE(IO2,2010) MAX
   21  WRITE(IO2,2070) EPSP
   22  WRITE(IO2,2020) EPSCNV
   23  WRITE(IO2,2030) EPSPMUL
   24  WRITE(IO2,2040) XSTART
   25  WRITE(IO2,2050) XEND
   26  IF(NP.GT.2) GO TO 90
   27  CALL QUAD(UP,VP,NP,UROOT,VROOT,MULTI,EPSP)
   28  85 WRITE(IO2,1037)
   29  WRITE(IO2,1086) (I,UROOT(I),VROOT(I),MULTI(I),I=1,NRROOT)
   30  GO TO 10
   31  90 CALL DERIV(UP,VP,NP,UDP,VDP,NDP)
   32  CALL GCD(UP,VP,NP,UDP,VDP,NDP,URK,VRK,NRK,EPSP)
   33  CALL DIVIDE(UP,VP,NP,URK,VRK,NRK,UH,VH,NH)
   34  KKK=NH+1
   35  WRITE(IO2,1060)
   36  WRITE(IO2,1035) (HNAME,I,UH(I),VH(I),I=1,KKK)
   37  IF(NH.GT.2) GO TO 150
   38  CALL QUAD(UH,VH,NH,UROOT,VROOT,NRROOT,MULTI,EPSP)
   39  CALL MULTIP(UP,VP,NP,UROOT,VROOT,NRROOT,MULTI,EPSPMUL)
   40  GO TO 85
   41  150 CALL METHOD(UH,VH,NH,UOIAPP,VOIAPP,NIAP,UROOT,VROOT,NRROOT,URIAPP,V

```

TABLE XVIII-B (Continued)

```

42     IRIAPP,UDPOLY,VDPOLY,MULTI,ND,IROOT,EPSQ,XSTART,XEND,EPSCNV,EPSMUL)
43     IF(NROOT.NE.0) GO TO 170
44     WRITE(IO2,1070) NOPOLY
45     GO TO 200
46 170 CALL MULTIP(UP,VP,NP,UROOT,VROOT,NROOT,MULTI,EPSMUL)
47     WRITE(IO2,1065)
48     WRITE(IO2,1080)
49     WRITE(IO2,1085) (I,UROOT(I),VROOT(I),MULTI(I),URIAPP(I),VRIAPP(I),
50     I=1,IROOT)
51     KKK=IROOT+1
52     IF(IROOT.LT.NROOT) WRITE(IO2,1086) (I,UROOT(I),VROOT(I),MULTI(I),I
53     I=KKK,NROOT)
54 200 IF(ND.EQ.0) GO TO 10
55 230 WRITE(IO2,1090)
56     WRITE(IO2,1035) (DNAME,J,UDPOLY(J),VDPOLY(J),J=1,ND)
57     GO TO 10
58 1000 FORMAT(3(I2,1X),9X,I3,1X,4(D6.0,1X),13X,2(D7.0,1X),I1)
59 1010 FORMAT(2D30.0)
60 1020 FORMAT(2D30.0)
61 1030 FORMAT(1H1,10X,41HGREATEST COMMON DIVISOR METHOD USED WITH ,2(A4),
62     135HMETHOD TO FIND ZEROS OF POLYNOMIALS/11X,18HPOLYNOMIAL NUMBER ,I
63     22,11H OF DEGREE ,I2///1X,28HTHE COEFFICIENTS OF P(X) ARE//)
64 1035 FORMAT(3X,A2,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
65 1037 FORMAT(///1X,13HZEROS OF P(X),55X,14HMULTIPLICITIES//)
66 1086 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,8X,23HS
67     IOLVED BY DIRECT METHOD)
68 1060 FORMAT(///1X,42HTHE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE//)
69 1070 FORMAT(///1X,42HNO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
70 1065 FORMAT(///1X,61HAFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
71     I OF P(X) ARE)
72 1085 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,7X,D23.
73     116,3H + ,D23.16,2H I)
74 1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
75     IZEROS WERE FOUND//)
76 1080 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
77     I APPROXIMATION//)
78 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,I2)
79 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
80 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
81 2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)
82 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
83 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
84 2060 FORMAT(///1X)
85 2070 FORMAT(1X,31HTEST FOR ZERO IN SUBROUTINE GCD,3X,D9.2)
86     END

```

TABLE XVIII-B (Continued)

```

77      SUBROUTINE METHOD(RA,VA,N,RXZERO,VXZERO,NIAP,RX,VX,K,RXINIT,VXINIT
      1,RD,VD,MULT,ND,IROOT,EPSQ,XSTART,XEND,EPSCNV,EPSMUL)
C      *****
C      *
C      * THIS SUBROUTINE USES NEWTON'S METHOD TO EXTRACT THE ZEROS OF A
C      * POLYNOMIAL. A SEQUENCE OF APPROXIMATIONS IS CONSTRUCTED CONVERGING TO A
C      * ZERO OF THE POLYNOMIAL BY USING THE ITERATION FORMULA
C      *
C      *      X(N+1) = X(N) - P(X(N))/P'(X(N)).
C      *
C      *****
78      DOUBLE PRECISION RA,VA,RXZERO,VXZERO,VB,VB,RCOEF,VCOEF,RX,VX,RXINI
      1T,VXINIT,RC,VC,RD,VD,RPX,VPX,RDPX,VDPX,RXNEW,VXNEW,RXO,VXO,EPSCHK,
      2EPSCNV,EPSQ,EPSMUL,XSTART,XEND,ABPX,ABDPX
79      DOUBLE PRECISION DSQRT
80      DIMENSION RA(26),VA(26),RB(26),VB(26),RC(26),VC(26),RD(26),VD(26),
      1RCOEF(26),VCOEF(26),MULT(25),RXZERO(25),VXZERO(25),RX(25),VX(25),R
      2XINIT(25),VXINIT(25)
81      COMMON EPSCHK,I02,MAX
82      IF(NIAP.NE.0) GO TO 1
83      NIAP=N
84      CALL GENAPP(RXZERO,VXZERO,NIAP,XSTART)
85      1 NA=N+1
86      NDEF=N
87      L=1
88      ITER=0
89      NROOT=0
90      IROOT=0
91      ITIME=0
92      ND=0
93      IALTER=0
94      K=0
95      RXO=RXZERO(L)
96      VXO=VXZERO(L)
97      DO 5 I=1,NA
98      RCOEF(I)=RA(I)
99      5 VCOEF(I)=VA(I)
100     10 CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,VB,VB,RC,VC,RPX,VPX,RDPX,VDPX
      1)
101     ABPX=DSQRT(RPX*RPX+VPX*VPX)
102     ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)
103     IF(ABDPX.NE.0.0) GO TO 20
104     IF(ABPX.EQ.0.0) GO TO 70
105     GO TO 110
106     20 CALL NEWTON(RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW)
107     ITER=ITER+1
108     RXO=RXNEW
109     VXO=VXNEW
110     EPSCHK=EPSCNV
111     CALL CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
112     IF(KANS.EQ.1) GO TO 70
113     IF(ITER.GE.MAX) GO TO 40
114     GO TO 10
115     40 CALL ALTER(RXZERO(L),VXZERO(L),IALTER,ITIME)
116     IF(IALTER.GT.5) GO TO 110
117     RXO=RXZERO(L)
118     VXO=VXZERO(L)
119     ITER=0
120     GO TO 10
121     60 ND=NDEF+1
122     DO 65 J=1,ND

```

TABLE XVIII-B (Continued)

123	RD(J)=RCOEF(J)	
124	65 VD(J)=VCOEF(J)	
125	GO TO 140	
126	70 NROOT=NROOT+1	268
127	K=K+1	272
128	MULT(K)=1	276
129	RX(K)=RXO	280
130	VX(K)=VXO	281
131	RXINIT(K)=RXZERO(L)	288
132	VXINIT(K)=VXZERO(L)	289
133	CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,ROPX,ROPX,1)	
134	80 IF(NROOT.GE.N) GO TO 147	
135	NDEF=NDEF-1	
136	NUM=NDEF+1	
137	DO 105 I=1,NUM	294
138	RCOEF(I)=RB(I)	296
139	VCOEF(I)=VB(I)	297
140	CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,ROPX,ROPX,1)	
141	ABPX=DSQRT(RPX*RPX+VPX*VPX)	
142	ABDPX=DSQRT(RDPX*ROPX+VDPX*VDPX)	
143	IF(ABDPX.NE.0.0) GO TO 107	
144	IF(ABPX.EQ.0.0) GO TO 130	
145	GO TO 110	
146	107 CONTINUE	
147	EPSCHK=EPSMUL	
148	CALL CHECK(RPX,VPX,ROPX,VDPX,RXO,VXO,KANS)	
149	IF(KANS.EQ.1) GO TO 130	300
150	110 IF(NDEF.GT.2) GO TO 113	
151	IROOT=K	
152	CALL QUAD(RCOEF,VCOEF,NDEF,RX,VX,K,MULT,EPSQ)	
153	GO TO 150	
154	113 IF(L.LT.NIAP) GO TO 115	
155	IF(XEND.EQ.0.0) GO TO 60	
156	IF(XSTART.GT.XEND) GO TO 60	
157	NIAP=N	
158	CALL GENAPP(RXZERO,VXZERO,NIAP,XSTART)	
159	L=0	
160	115 L=L+1	
161	RXO=RXZERO(L)	312
162	VXO=VXZERO(L)	313
163	ITER=0	316
164	IALTER=0	320
165	GO TO 10	324
166	130 MULT(K)=MULT(K)+1	328
167	NROOT=NROOT+1	332
168	GO TO 80	336
169	140 IF(K.EQ.0) GO TO 160	338
170	147 IROOT=K	
171	150 WRITE(102,1025)	
172	WRITE(102,1050)	
173	WRITE(102,1060) (I,RX(I),VX(I),MULT(I),RXINIT(I),VXINIT(I),I=1,IROT)	
174	10T)	
174	KKK=IROOT+1	
175	IF(IROOT.LT.K) WRITE(102,1062) (I,RX(I),VX(I),MULT(I),I=KKK,K)	
176	EPSCHK=EPSCNV	
177	CALL BETTER(K,RXZERO,VXZERO,RX,VX,NA,RA,VA,RCOEF,VCOEF,N,RC,VC,VB,1VB)	
178	160 RETURN	

TABLE XVIII-B (Continued)

```

179 1025 FORMAT(///1X,61HBEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
      10F P(X) ARE)
180 1050 FORMAT(///2X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
      1 APPROXIMATION//)
181 1060 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,7X,D23.
      116,3H + ,D23.16,2H I)
182 1062 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,8X,23HS
      10LVED BY DIRECT METHOD)
183      END

```

450

```

184      SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
C      *****
C      *
C      * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE
C      * DEGREE OF THE ORIGINAL POLYNOMIAL.
C      *
C      *****
185      DOUBLE PRECISION APPR,APPI,XSTART,DUMMY,BETA
186      DOUBLE PRECISION DCOS,DSIN
187      DIMENSION APPR(25),APPI(25)
188      COMMON DUMMY,I02,MAX
189      IF(XSTART.EQ.0.0) XSTART=0.5
190      BETA=0.2617994
191      DO 10 I=1,NAPP
192      APPR(I)=XSTART*DCOS(BETA)
193      APPI(I)=XSTART*DSIN(BETA)
194      BETA=BETA+0.5235988
195      10 XSTART=XSTART+0.5
196      RETURN
197      END

```

TABLE XVIII-B (Continued)

```

198      SUBROUTINE ALTER(XOLDR,XOLDI,NALTER,ITIME)
C      *****
C      *
C      * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO
C      * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.
C      *
C      *****
199      DOUBLE PRECISION XOLDR,XOLDI,DUMMY,ABXOLD,BETA
200      DOUBLE PRECISION DCOS,DSIN
201      DOUBLE PRECISION DATAN2
202      DOUBLE PRECISION DSQRT
203      COMMON DUMMY,IO2,MAX
204      IF(ITIME.NE.0) GO TO 5
205      ITIME =1
206      WRITE(IO2,1010) MAX
207      5 IF(NALTER.EQ.0) GO TO 10
208      WRITE(IO2,1000) XOLDR,XOLDI
209      GO TO 20
210      10 ABXOLD=DSQRT(XOLDR*XOLDR+XOLDI*XOLDI)
211      BETA=DATAN2(XOLDI,XOLDR)
212      WRITE(IO2,1020) XOLDR,XOLDI
213      20 NALTER=NALTER+1
214      IF(NALTER.GT.5) RETURN
215      GO TO (30,40,30,40,30),NALTER
216      30 XOLDR=-XOLDR
217      XOLDI=-XOLDI
218      GO TO 50
219      40 BETA=BETA+1.0471976
220      XOLDR=ABXOLD*DCOS(BETA)
221      XOLDI=ABXOLD*DSIN(BETA)
222      50 RETURN
223      1000 FORMAT(1X,D23.16,3H + ,D23.16,2H I,10X,21HALTERED APPROXIMATION)
224      1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
225      1020 FORMAT(/1X,D23.16,3H + ,D23.16,2H I,10X,21HINITIAL APPROXIMATION)
226      END

```


TABLE XVIII-B (Continued)

```

227 SUBROUTINE QUAD(UA,VA,NA,URROOT,VROOT,NROOT,MULTI,EPST)
C *****
C *
C * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
C * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE *
C * QUADRATIC IS DONE USING THE QUADRATIC FORMULA. *
C *
C *****
228 DOUBLE PRECISION UA,VA,URROOT,VROOT,BBB,UAAA,VAAA,UDISC,VDISC,UDUMM
    LY,VDUMMY,RDUMMY,SDUMMY,EPST,UBBB,VBBB
229 DOUBLE PRECISION DSQRT
230 DIMENSION UA(26),VA(26),URROOT(25),VROOT(25),MULTI(25)
231 IF(NA.EQ.2) GO TO 7
232 IF(NA.EQ.1) GO TO 5
233 URROOT(NROOT+1)=0.0
234 VROOT(NROOT+1)=0.0
235 MULTI(NROOT+1)=1
236 NROOT=NROOT+1
237 GO TO 50
238 5 BBB=UA(1)*UA(1)+VA(1)*VA(1)
239 URROOT(NROOT+1)=(-UA(2)*UA(1)-VA(2)*VA(1))/BBB
240 VROOT(NROOT+1)=(-VA(2)*UA(1)+UA(2)*VA(1))/BBB
241 MULTI(NROOT+1)=1
242 NROOT=NROOT+1
243 GO TO 50
244 7 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-4.0*(UA(1)*UA(3)-VA(1)*VA(3))
245 VDISC=(VA(2)*UA(2)+UA(2)*VA(2))-4.0*(VA(1)*UA(3)+UA(1)*VA(3))
246 BBB=DSQRT(UDISC*UDISC+VDISC*VDISC)
247 IF(BBB.LT.EPST) GO TO 10
248 CALL COMSQT(UDISC,VDISC,UDUMMY,VDUMMY)
249 UBBB=-UA(2)+UDUMMY
250 VBBB=-VA(2)+VDUMMY
251 RDUMMY=-UA(2)-UDUMMY
252 SDUMMY=-VA(2)-VDUMMY
253 UAAA=2.0*UA(1)
254 VAAA=2.0*VA(1)
255 BBB=UAAA*UAAA+VAAA*VAAA
256 URROOT(NROOT+1)=(UBBB*UAAA+VBBB*VAAA)/BBB
257 VROOT(NROOT+1)=(VBBB*UAAA-UBBB*VAAA)/BBB
258 URROOT(NROOT+2)=(RDUMMY*UAAA+SDUMMY*VAAA)/BBB
259 VROOT(NROOT+2)=(SDUMMY*UAAA-RDUMMY*VAAA)/BBB
260 MULTI(NROOT+1)=1
261 MULTI(NROOT+2)=1
262 NROOT=NROOT+2
263 GO TO 50
264 10 UAAA=2.0*UA(1)
265 VAAA=2.0*VA(1)
266 BBB=UAAA*UAAA+VAAA*VAAA
267 URROOT(NROOT+1)=(-UA(2)*UAAA-VA(2)*VAAA)/BBB
268 VROOT(NROOT+1)=(-VA(2)*UAAA+UA(2)*VAAA)/BBB
269 MULTI(NROOT+1)=2
270 NROOT=NROOT+1
271 50 RETURN
272 END

```

TABLE XVIII-B (Continued)

```

273     SUBROUTINE HORNER(RA,VA,RXO,VXO,NDEF,VB,RC,VC,RPX,VPX,RDPX,VDPX
      1)
      C *****
      C *
      C * HORNER'S METHOD COMPUTES THE VALUE OF A POLYNOMIAL P(X) AT A POINT D AND *
      C * ITS DERIVATIVE AT D. SYNTHETIC DIVISION IS USED TO DEFLATE THE *
      C * POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D). *
      C *
      C *****
274     DOUBLE PRECISION VDPX,RXO,VXO,VB,RC,VC,RPX,VPX,RDPX,RA,VA
275     DIMENSION RA(26),VA(26),RB(26),VB(26),RC(26),VC(26)
276     RB(1)=RA(1)
277     VB(1)=VA(1)
278     NUM=NDEF+1
279     DO 10 I=2,NUM
280     RB(I)=RA(I)+(RB(I-1)*RXO-VB(I-1)*VXO)
281     10 VB(I)=VA(I)+(VB(I-1)*RXO+RB(I-1)*VXO)
282     RPX=RB(NUM)
283     VPX=VB(NUM)
284     RC(1)=RB(1)
285     VC(1)=VB(1)
286     IF(NDEF.LT.2) GO TO 25
287     DO 20 J=2,NDEF
288     RC(J)=RB(J)+(RC(J-1)*RXO-VC(J-1)*VXO)
289     20 VC(J)=VB(J)+(VC(J-1)*RXO+RC(J-1)*VXO)
290     25 RDPX=RC(NDEF)
291     VDPX=VC(NDEF)
292     RETURN
293     END

```

516
517
520
524
532
533
540
541
544
552
553
572
580


```

294     SUBROUTINE NEWTON(RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW)
      C *****
      C *
      C * THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-
      C * IMATION BY USING THE ITERATION FORMULA
      C *
      C *
      C *
      C *
      C *****
295     DOUBLE PRECISION RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW,ARG
296     DOUBLE PRECISION DSQRT
297     DOUBLE PRECISION DDD
298     ARG=RDPX*RDPX+VDPX*VDPX
299     DDD=DSQRT(ARG)
300     IF(DDD.EQ.0.0) RETURN
301     RXNEW=RXO-((RPX*RDPX+VPX*VDPX)/ARG)
302     VXNEW=VXO-((VPX*RDPX-RPX*VDPX)/ARG)
303     RETURN
304     END

```

600
616
620

TABLE XVIII-B (Continued)

```

305     SUBROUTINE CHECK(RPX,VPX, RDPX, VDPX, RXO, VXO, KANS)
C     *****
C     *
C     * THIS SUBROUTINE CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
C     * IMATIONS BY TESTING THE EXPRESSION
C     * ABSOLUTE VALUE OF {P(X(N))/P'(X(N))}/ABSOLUTE VALUE OF X(N+1).
C     * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
C     *
C     *****
306     DOUBLE PRECISION RPX,VPX, RDPX, VDPX, RXO, VXO, ABSXO, ABSQUO, RDUMMY, VDU
      1MMY, EPS
      749
307     DOUBLE PRECISION DDD
308     DOUBLE PRECISION ARG
309     DOUBLE PRECISION DSQRT
310     COMMON EPS, I02, MAX
      751
311     ABSXO=DSQRT(RXO*RXO+VXO*VXO)
312     IF(ABSXO.EQ.0.) GO TO 25
313     ARG= RDPX* RDPX+ VDPX* VDPX
314     DDD=DSQRT(ARG)
315     IF(DDD.EQ.0.) GO TO 25
316     RDUMMY=(RPX* RDPX+ VPX* VDPX)/ARG
317     VDUMMY=(VPX* RDPX- RPX* VDPX)/ARG
318     ABSQUO=DSQRT(RDUMMY* RDUMMY+ VDUMMY* VDUMMY)
319     IF(ABSQUO/ABSXO.LT.EPS) GO TO 10
320     KANS=0
      760
321     RETURN
      764
322     10 KANS=1
      768
323     RETURN
      772
324     25 KANS=0
325     RETURN
326     END
      780

```

TABLE XVIII-B (Continued)

```

327     SUBROUTINE BETTER(K,RXZERO,VXZERO,RX,VX,NA,RA,VA,RCOEF,VCOEF,N,RC,      800
      1VC,VB,VB)
C     *
C     *
C     * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND *
C     * BY USING THEM AS INITIAL APPROXIMATIONS WITH NEWTON'S METHOD APPLIED TO *
C     * THE FULL, UNDEFLATED POLYNOMIAL. *
C     *
C     *
328     DOUBLE PRECISION RXZERO,VXZERO,RX,VX,RA,VA,RCOEF,VCOEF,RC,VC,VB,VB      805
      1,RXO,VXO,RPX,VPX,ROPX,ROPX,ROPX,ROPX,RXNEW,VXNEW,EPS                    806
329     DOUBLE PRECISION ABPX,ABDPX
330     DOUBLE PRECISION DSQRT
331     DIMENSION RXZERO(25),VXZERO(25),RX(25),VX(25),RA(26),VA(26),RCOEF(      808
      126),VCOEF(26),RC(26),VC(26),VB(26),VB(26)
332     COMMON EPS,IO2,MAX
333     DO 10 I=1,K                                                                812
334     RXZERO(I)=RX(I)                                                            815
335     10 VXZERO(I)=VX(I)                                                         816
336     DO 20 I=1,NA
337     RCOEF(I)=RA(I)                                                            824
338     20 VCOEF(I)=VA(I)                                                         825
339     DO 50 J=1,K                                                                828
340     RXO=RXZERO(J)                                                            832
341     VXO=VXZERO(J)                                                            833
342     NN=N                                                                      834
343     ITER=0                                                                    836
344     30 CALL HORNER(RCOEF,VCOEF,RXO,VXO,NN,VB,VB,RC,VC,RPX,VPX,ROPX,ROPX)
345     ABPX=DSQRT(RPX*RPX+VPX*VPX)
346     ABDPX=DSQRT(ROPX*ROPX+ROPX*ROPX)
347     IF(ABDPX.NE.0.0) GO TO 33
348     IF(ABPX.EQ.0.0) GO TO 40
349     GO TO 34
350     33 CALL NEWTON(RPX,VPX,ROPX,ROPX,RXO,VXO,RXNEW,VXNEW)
351     ITER=ITER+1                                                                856
352     RXO=RXNEW                                                                860
353     VXO=VXNEW                                                                861
354     CALL CHECK(RPX,VPX,ROPX,ROPX,RXO,VXO,KANS)
355     IF(KANS.EQ.1) GO TO 40                                                    844
356     IF(ITER.GE.MAX) GO TO 35
357     GO TO 30                                                                  864
358     34 WRITE(IO2,1112) RXO,VXO
359     35 WRITE(IO2,100) J,RXZERO(J),VXZERO(J)
360     WRITE(IO2,200) MAX
361     40 RX(J)=RXO                                                                870
362     VX(J)=VXO                                                                871
363     50 CONTINUE                                                                872
364     RETURN                                                                    876
365     1112 FORMAT(1H0,36H THE VALUE OF THE DERIVATIVE AT XO = ,D23.16,3H + ,D2
      13.16,2H I,10H IS ZERO.)
366     100 FORMAT(42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,D23
      1.16,3H + ,D23.16,2H I,18H DID NOT CONVERGE.)
367     200 FORMAT(33H THE PRESENT APPROXIMATION AFTER ,I3,29H ITERATIONS IS P
      1RINTED BELOW.)
368     END

```

TABLE XVIII-B (Continued)

```

369      SUBROUTINE COMSQT(UX,VX,UY,VY)
C      *****
C      *
C      * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
C      *
C      *****
370      DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
371      DOUBLE PRECISION DSQRT,DABS
372      R=DSQRT(UX*UX+VX*VX)
373      AAA=DSQRT(DABS((R+UX)/2.0))
374      BBB=DSQRT(DABS((R-UX)/2.0))
375      IF(VX) 10,20,30
376      10 UY=AAA
377         VY=-1.0*BBB
378         GO TO 100
379      20 IF(UX) 40,50,60
380      30 UY=AAA
381         VY=BBB
382         GO TO 100
383      40 DUMMY=DABS(UX)
384         UY=0.0
385         VY=DSQRT(DUMMY)
386         GO TO 100
387      50 UY=0.0
388         VY=0.0
389         GO TO 100
390      60 DUMMY=DABS(UX)
391         UY=DSQRT(DUMMY)
392         VY=0.0
393      100 RETURN
394      END

395      SUBROUTINE DERIV(UA,VA,NA,UB,VB,NB)
C      *****
C      *
C      * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF
C      * ITS DERIVATIVE P'(X).
C      *
C      *****
396      DOUBLE PRECISION UA,VA,UB,VB
397      DIMENSION UA(26),VA(26),UB(26),VB(26)
398      IF(NA.GE.2) GO TO 30
399      IF(NA.EQ.1) GO TO 20
400      UB(1)=0.0
401      VR(1)=0.0
402      NB=-1
403      GO TO 50
404      20 UB(1)=UA(1)
405         VB(1)=VA(1)
406         NR=0
407         GO TO 50
408      30 NB=NA-1
409         DO 40 I=1,NA
410            BBB=NA-I+1
411            UB(I)=BBB*UA(I)
412            VB(I)=BBB*VA(I)
413      40 RETURN
414      END

```

TABLE XVIII-B (Continued)

```

415      SUBROUTINE DIVIDE(UF,VF,NF,UG,VG,NG,UH,VH,NH)
C      *****
C      *
C      * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE
C      * QUOTIENT POLYNOMIAL H(X) = F(X)/G(X).
C      *
C      *****
416      DOUBLE PRECISION UF,VF,UG,VG,UH,VH,UX,VX,UC,VC,UPX,VPX,UDPX,VDPX,U
1TERM,VTERM,UDUMMY,VDUMMY,EPS
      DOUBLE PRECISION DENOM
417      DIMENSION UF(26),VF(26),UG(26),VG(26),UH(26),VH(26),UC(26),VC(26)
418      COMMON EPS,IO2,MAX
419      IF(NG.GE.2) GO TO 60
420      IF(NG.EQ.1) GO TO 30
421      NH=NF
422      KKK=NF+1
423      DO 20 I=1,KKK
424      DENOM=UG(I)*UG(I)+VG(I)*VG(I)
425      UH(I)=(UF(I)*UG(I)+VF(I)*VG(I))/DENOM
426      VH(I)=(VF(I)*UG(I)-UF(I)*VG(I))/DENOM
427      20 GO TO 100
428      30 UDUMMY=-1.0*UG(2)
429      VDUMMY=-1.0*VG(2)
430      DENOM=UG(1)*UG(1)+VG(1)*VG(1)
431      UX=(UDUMMY*UG(1)+VDUMMY*VG(1))/DENOM
432      VX=(VDUMMY*UG(1)-UDUMMY*VG(1))/DENOM
433      NNF=NF
434      CALL HORNER(UF,VF,UX,VX,NNF,UH,VH,UC,VC,UPX,VPX,UDPX,VDPX)
435      NH=NNF-1
436      GO TO 100
437      60 NH=NF-NG
438      DENOM=UG(1)*UG(1)+VG(1)*VG(1)
439      UH(1)=(UF(1)*UG(1)+VF(1)*VG(1))/DENOM
440      VH(1)=(VF(1)*UG(1)-UF(1)*VG(1))/DENOM
441      KKK=NH+1
442      DO 95 I=2,KKK
443      UTERM=UF(I)
444      VTERM=VF(I)
445      J=2
446      K=I-1
447      70 UTERM=UTERM-(UG(J)*UH(K)-VG(J)*VH(K))
448      VTERM=VTERM-(VG(J)*UH(K)+UG(J)*VH(K))
449      IF(J.EQ.I) GO TO 90
450      IF(J.GE.NG+1) GO TO 90
451      J=J+1
452      K=K-1
453      GO TO 70
454      90 DENOM=UG(1)*UG(1)+VG(1)*VG(1)
455      UH(I)=(UTERM*UG(1)+VTERM*VG(1))/DENOM
456      95 VH(I)=(VTERM*UG(1)-UTERM*VG(1))/DENOM
457      100 RETURN
458      END
459

```

TABLE XVIII-B (Continued)

```

460      SUBROUTINE GCD(UP,VP,NP,UDP,VDP,NDP,URK,VRK,NK,EPG)
C      *****
C      *
C      * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG.
C      * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND
C      * DP(X).
C      *
C      *****
461      DOUBLE PRECISION EPG,UP,VP,UDP,VDP,NDP,URK,VRK,UR1,VR1,UR2,VR2,UT,VT,
      1UU,VU,BBB,EPG
462      DOUBLE PRECISION DENOM
463      DOUBLE PRECISION DSQRT
464      DIMENSION UR1(26),VR1(26),UR2(26),VR2(26),UT(26),VT(26),URK(26),VR
      1K(26),UP(26),VP(26),UDP(26),VDP(26)
465      COMMON EPG,IO2,MAX
466      N1=NP
467      N2=NDP
468      KKK=N1+1
469      DO 10 I=1,KKK
470      UR1(I)=UP(I)
471      10 VR1(I)=VP(I)
472      KKK=N2+1
473      DO 20 I=1,KKK
474      UR2(I)=UDP(I)
475      20 VR2(I)=VDP(I)
476      50 DENOM=UR2(1)*UR2(1)+VR2(1)*VR2(1)
477      UU=(UR1(1)*UR2(1)+VR1(1)*VR2(1))/DENOM
478      VU=(VR1(1)*UR2(1)-UR1(1)*VR2(1))/DENOM
479      KKK=N2+1
480      DO 70 I=1,KKK
481      UT(I)=UR1(I)-(UU*UR2(I)-VU*VR2(I))
482      70 VT(I)=VR1(I)-(VU*UR2(I)+UU*VR2(I))
483      UT(1)=0.0
484      VT(1)=0.0
485      IF(N1.EQ.N2) GO TO 90
486      KKK=N1+1
487      NNN=N2+2
488      DO 80 I=NNN,KKK
489      UT(I)=UR1(I)
490      80 VT(I)=VR1(I)
491      90 KKK=N1+1
492      DO 100 I=1,KKK
493      BBB=DSQRT(UT(I)*UT(I)+VT(I)*VT(I))
494      IF(BBB.GT.EPG) GO TO 120
495      100 CONTINUE
496      NK=N2
497      KKK=NK+1
498      DO 110 I=2,KKK
499      DENOM=UR2(1)*UR2(1)+VR2(1)*VR2(1)
500      URK(I)=(UR2(I)*UR2(1)+VR2(I)*VR2(1))/DENOM
501      110 VRK(I)=(VR2(I)*UR2(1)-UR2(I)*VR2(1))/DENOM
502      URK(1)=1.0
503      VRK(1)=0.0
504      GO TO 200
505      120 M=N1+1-I
506      IF(M.LT.N2) GO TO 160
507      N1=M
508      KKK=N1+1
509      NNN=I-1
510      DO 150 J=1,KKK

```

TABLE XVIII-B (Continued)

```

511      UR1(J)=UT(NNN+J)
512  150  VR1(J)=VT(NNN+J)
513      GO TO 50
514  160  N1=N2
515      KKK=N1+1
516      DO 170 J=1,KKK
517      UR1(J)=UR2(J)
518  170  VR1(J)=VR2(J)
519      N2=M
520      KKK=N2+1
521      NNN=I-1
522      DO 180 J=1,KKK
523      UR2(J)=UT(NNN+J)
524  180  VR2(J)=VT(NNN+J)
525      GO TO 50
526  200  RETURN
527      END

528      SUBROUTINE MULTIP(UA,VA,NA,UROOT,VROOT,NR,MULTI,EPSPX)
C      *****
C      *
C      * GIVEN NR ZEROS OF A POLYNOMIAL, SUBROUTINE MULTIP COMPUTES THEIR
C      * MULTIPLICITIES.
C      *
C      *****
529      DOUBLE PRECISION UA,VA,UROOT,VROOT,EPSPX,UWORK,VWORK,UB,VB,UC,VC,U
1PX,VPX,UDPX,VDPX,BBB,EPS
530      DOUBLE PRECISION DSQRT
531      DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),UWORK(26),VWORK(26),MU
1LTI(25),UB(26),VB(26),UC(26),VC(26)
532      COMMON EPS,IO2,MAX
533      DO 50 I=1,NR
534      MULTI(I)=0
535      KKK=NA+1
536      DO 20 J=1,KKK
537      UWORK(J)=UA(J)
538  20  VWORK(J)=VA(J)
539      NWORK=NA
540  25  CALL HORNER(UWORK,VWORK,UROOT(I),VROOT(I),NWORK,UB,VB,UC,VC,UPX,VP
1X,UDPX,VDPX)
541      NWORK=NWORK-1
542      BBB=DSQRT(UPX*UPX+VPX*VPX)
543      IF(BBB.GT.EPSPX) GO TO 45
544      MULTI(I)=MULTI(I)+1
545      IF(NWORK.LT.1) GO TO 50
546      KKK=NWORK+1
547      DO 40 J=1,KKK
548      UWORK(J)=UB(J)
549  40  VWORK(J)=VB(J)
550      GO TO 25
551  45  IF(MULTI(I).EQ.0) WRITE(IO2,1000) EPSPX,I,UROOT(I),VROOT(I)
552  50  CONTINUE
553  100 RETURN
554  1000 FORMAT(///14H THE EPSILON (,D10.03,49H) CHECK IN SUBROUTINE MULTIP
1 INDICATES THAT ROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I/80H IS NO
2T CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIP
3LICITY 0//)
555      END

```

\$ENTRY

APPENDIX E

G.C.D. - MULLER'S METHOD

1. Use of the Programs

Two programs using the greatest common divisor method supported by Muller's method are presented here. The first is the single precision program. The second program is in double precision and is designed to perform double precision complex arithmetic. These programs are written for use on any computer using FORTRAN IV language. They have been tested on the IBM S/360 mod. 50 computer which has a 32 bit word. However, it may be necessary to change the system functions as described below. The single precision program may be changed to double precision as described below.

After selecting the desired program, the input data should be prepared as described in section 2.

Each program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree N where $N > 25$, certain array dimensions must be changed. These are listed in Table XIX.

TABLE XIX

PROGRAM CHANGES FOR SOLVING POLYNOMIALS OF DEGREE
GREATER THAN 25 BY G.C.D. - MULLER'S METHOD

Single PrecisionDouble Precision

Main Program

P(N+1)	UP(N+1), VP(N+1)
OIAPP(N)	UOIAPP(N), VOIAPP(N)
ROOT(N)	UROOT(N), VROOT(N)
MULTI(N)	MULTI(N)
DP(N+1)	UDP(N+1), VDP(N+1)
RK(N+1)	URK(N+1), VRK(N+1)
H(N+1)	UH(N+1), VH(N+1)

Subroutine MULTIP

A(N+1)	UA(N+1), VA(N+1)
ROOT(N)	UROOT(N), VROOT(N)
WORK(N+1)	UWORK(N+1), VWORK(N+1)
MULTI(N)	MULTI(N)
B(N+1)	UB(N+1), VB(N+1)

Subroutines DERIV, DIVIDE, and GCD

See corresponding subroutines of Table XIV in Appendix D.

Subroutines MULLER, BETTER, GENAPP,
HORNER and QUAD

See Main Program or corresponding subprograms of Table IX in Appendix C.

Certain computers may require that the system functions of Table X in Appendix C be changed in both the single precision and double precision programs.

When used on the IBM S/360 with the WATFOR compiler for FORTRAN IV, the system functions given in Tables X-A of Appendix C and XIV-A of Appendix D must be typed in a declaration statement. These also appear in the program listing. For use without the WATFOR compiler or on

other computers, these system functions might have to be removed.

The single precision program may be converted to double precision for use on machines equipped to perform double precision complex arithmetic provided the following changes or their equivalent are made and the system functions of Table XX are used and typed in a declaration statement where necessary. The changes presented below are those required for the IBM S/360. A "c" denotes a complex number and an "r" denotes a real number. The format statements should be changed from E-type to D-type.

In the main program and each subprogram change COMPLEX C_1, C_2, \dots to COMPLEX*16 C_1, C_2, \dots and add IMPLICIT REAL*8(A-H, O-Z).

TABLE XX

SYSTEM FUNCTIONS FOR CONVERTING SINGLE PRECISION G.C.D. -
MULLER'S METHOD TO DOUBLE PRECISION

<u>Single Precision</u>	changed to	<u>Double Precision</u>
	Subroutines GCD, MULTIP, MULLER, TEST, and QUAD	
CABS(c)	- absolute value -	CDABS(c)
	Subroutines CALC, GENAPP, ALTER, and QUAD	

See corresponding subprograms of Table XI in Appendix C.

2. Input Data for G.C.D. - Muller's Method

The input data for G.C.D. - Muller's method is prepared exactly as

that for G.C.D. - Newton's method as described in Appendix D, § 2.

3. Variables Used in G.C.D. - Muller's Method

The definitions of the major variables used in the G.C.D. - Muller's Method are referenced in Table XXI. The symbols used in the referenced tables are described in Appendix B, § 3. For variables not listed see the corresponding subprogram of Table VIII in Appendix B.

TABLE XXI

VARIABLES USED IN G.C.D. - MULLER'S METHOD

Main Program and Subroutines DERIV,
DIVIDE, GCD, and MULTIP

See corresponding subprogram of Table XVIII in Appendix D.

Subroutines MULLER, BETTER, TEST,
ALTER, HORNER, and QUAD

See Main Program or corresponding subprogram of Table XIII in Appendix C.

4. Description of Program Output

The output from the G.C.D. - Muller's method is the same as that for G.C.D. - Newton's method as described in Appendix D, § 4.

5. Informative and Error Messages

The informative and error messages are the same as those for G.C.D. - Newton's method as described in Appendix D, § 5.

MAIN PROGRAM

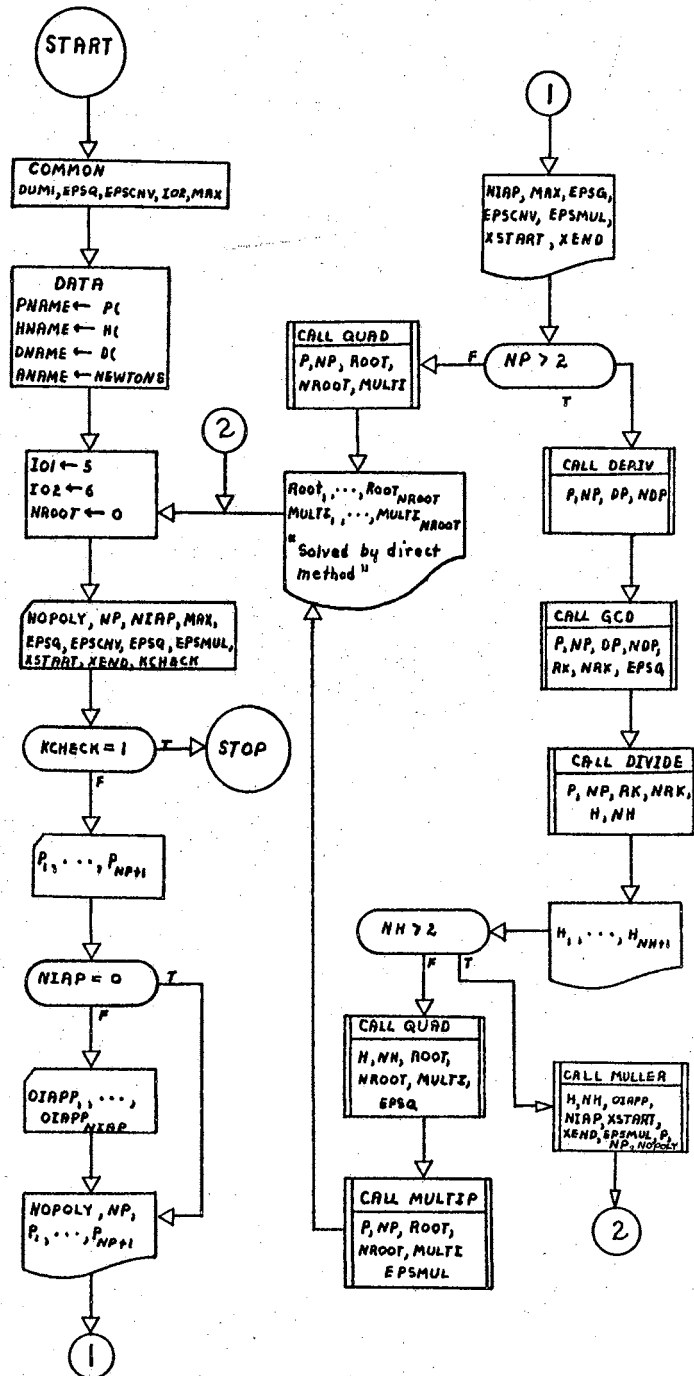


Figure 13.2. Flow Charts for G.C.D. - Muller's Method

MULLER

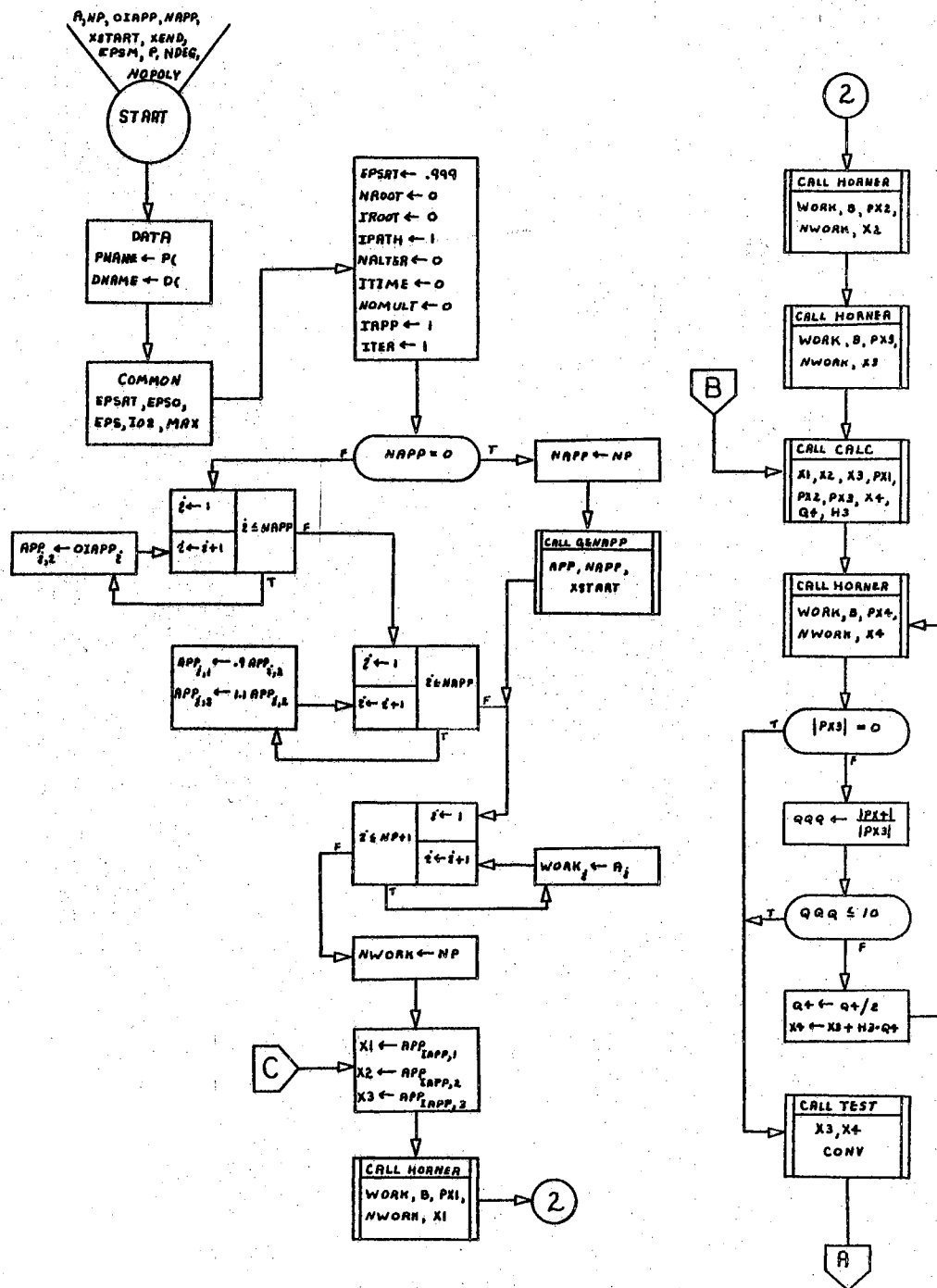


Figure 13.2. (Continued)

GCD

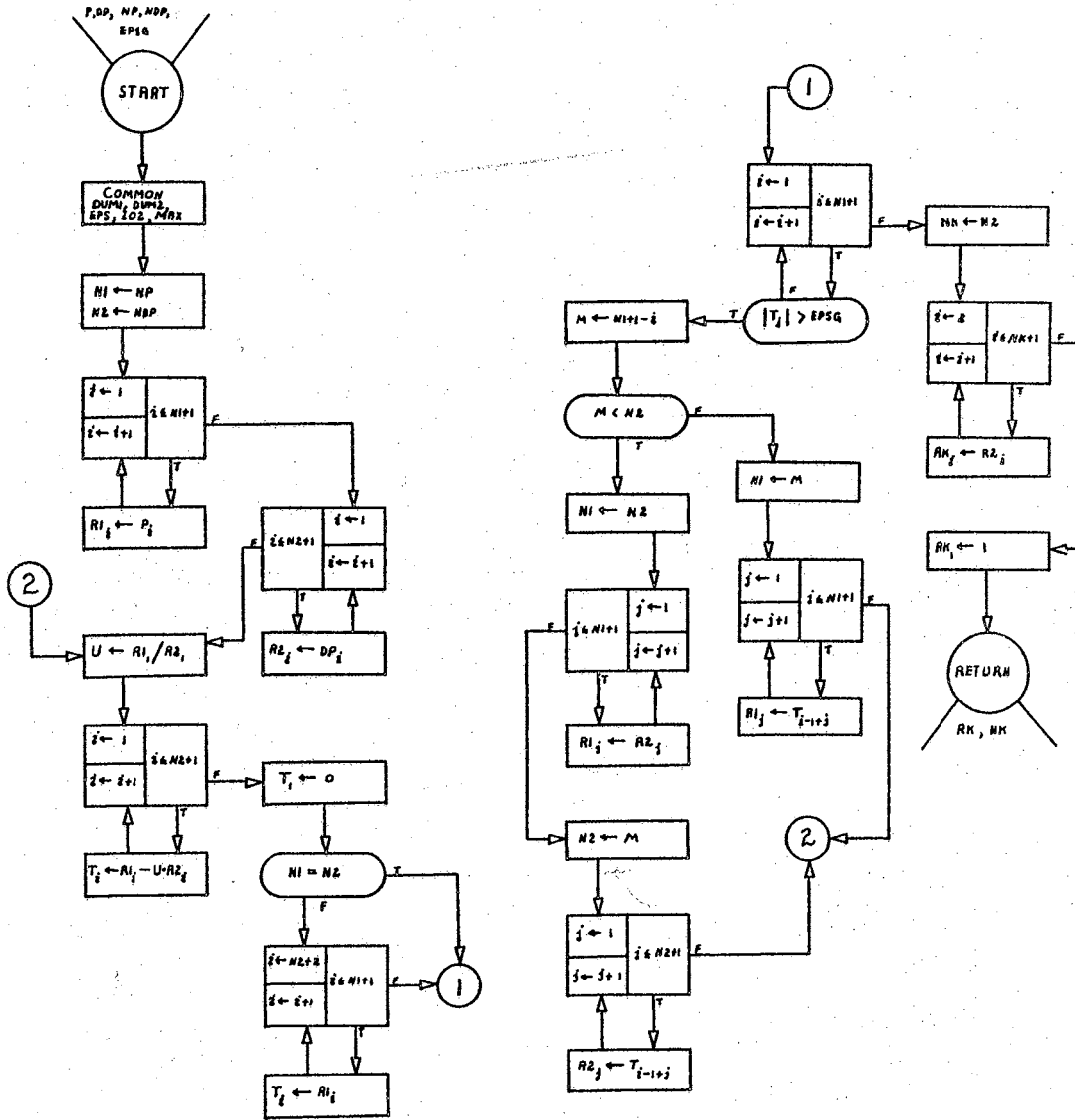


Figure 13.2. (Continued)

BETTER

HORNER

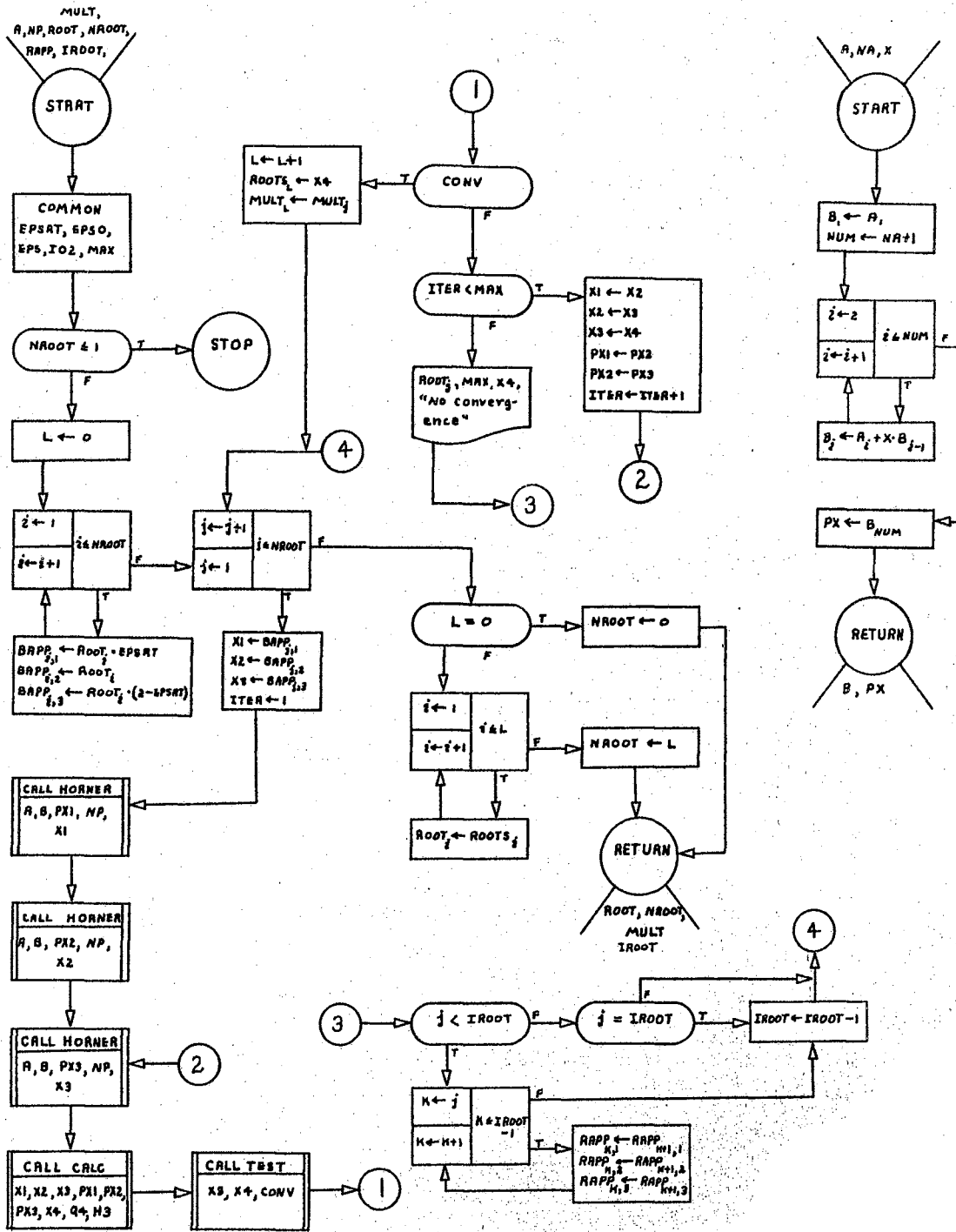
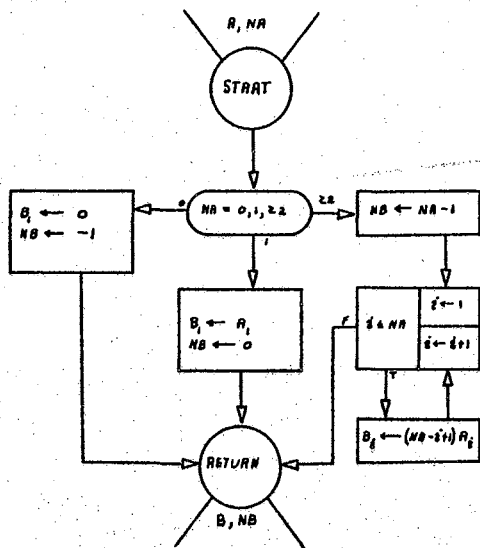


Figure 13.2. (Continued)

DERIV



MULTIP

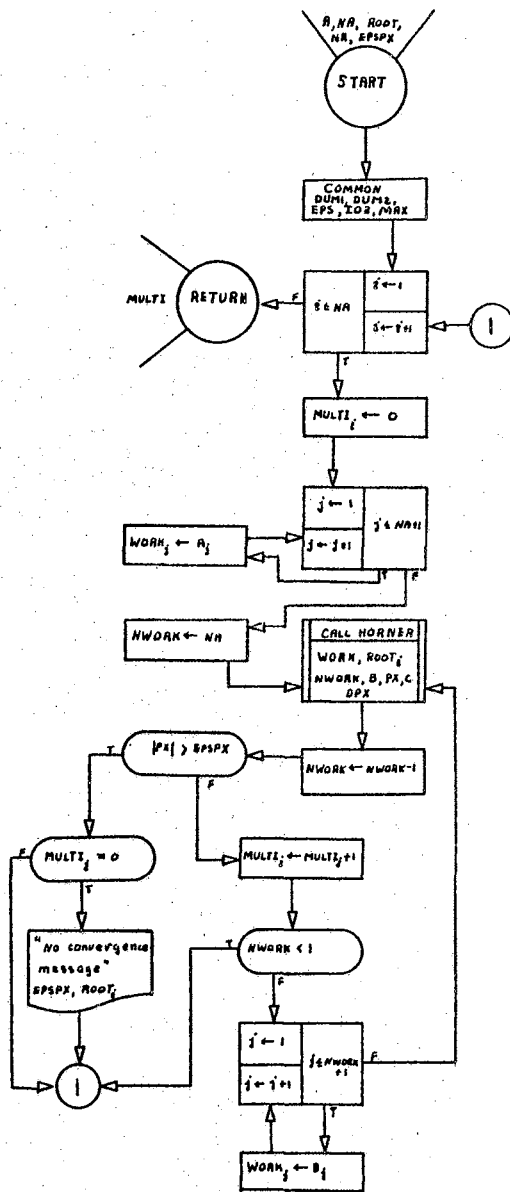
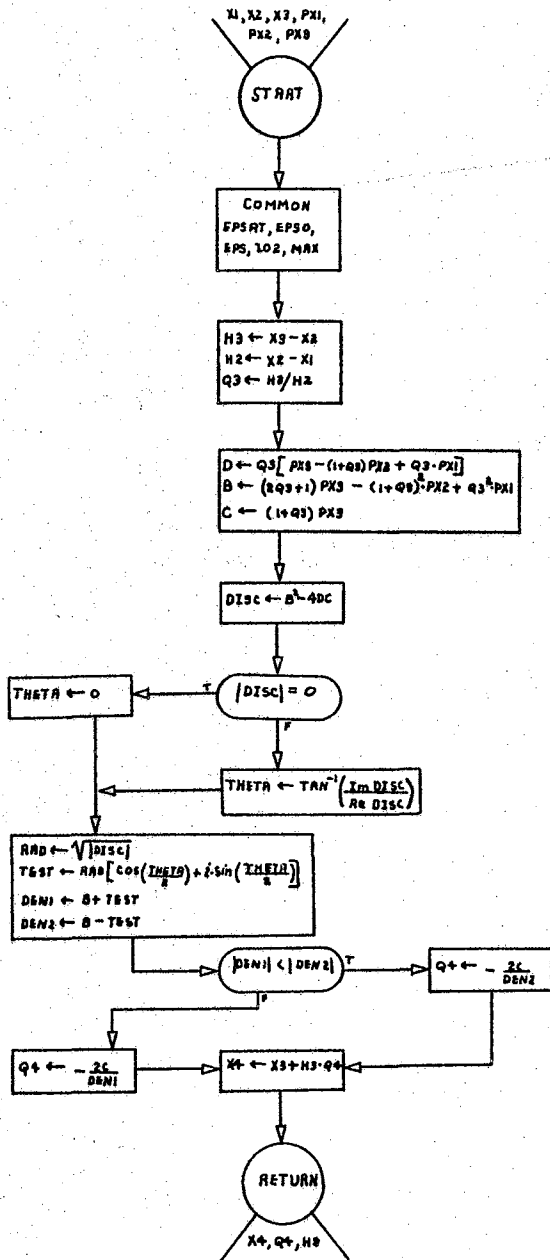


Figure 13.2. (Continued)

CALC



TEST

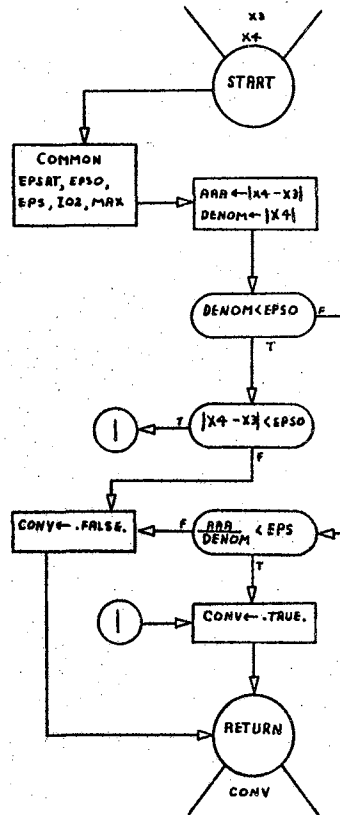
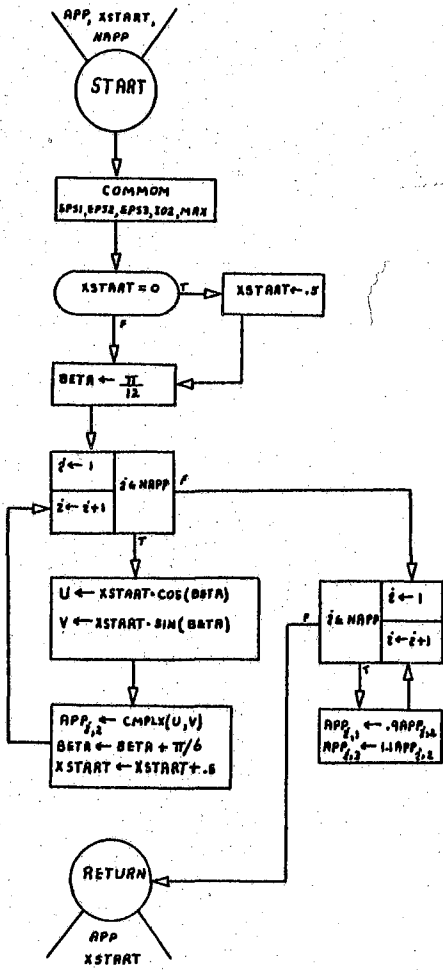


Figure 13.2. (Continued)

GENAPP



ALTER

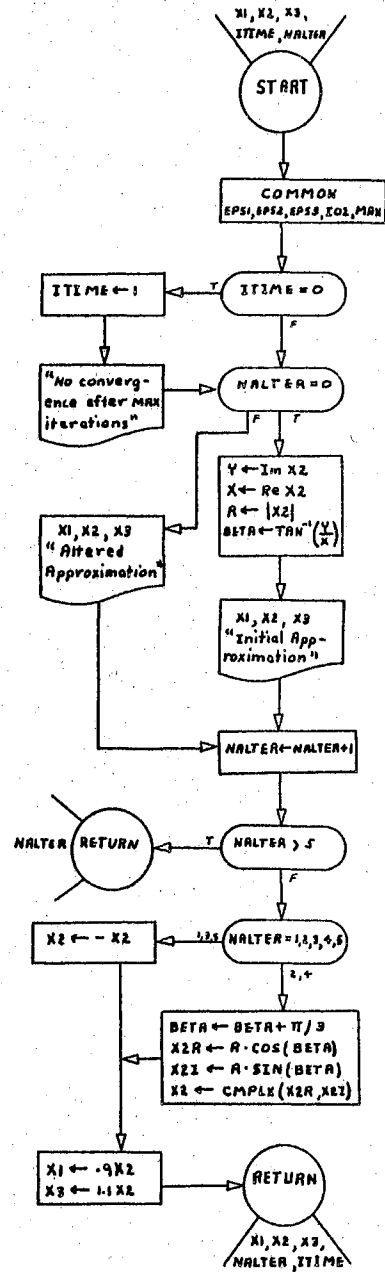


Figure 13.2. (Continued)

DIVIDE

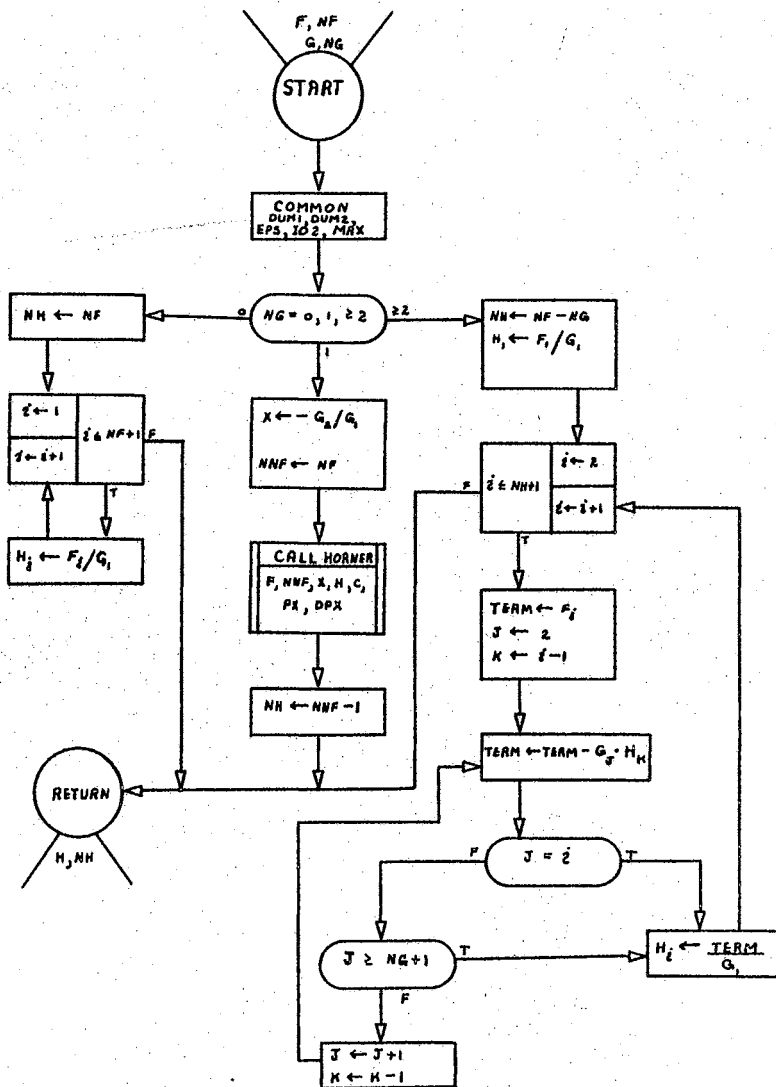
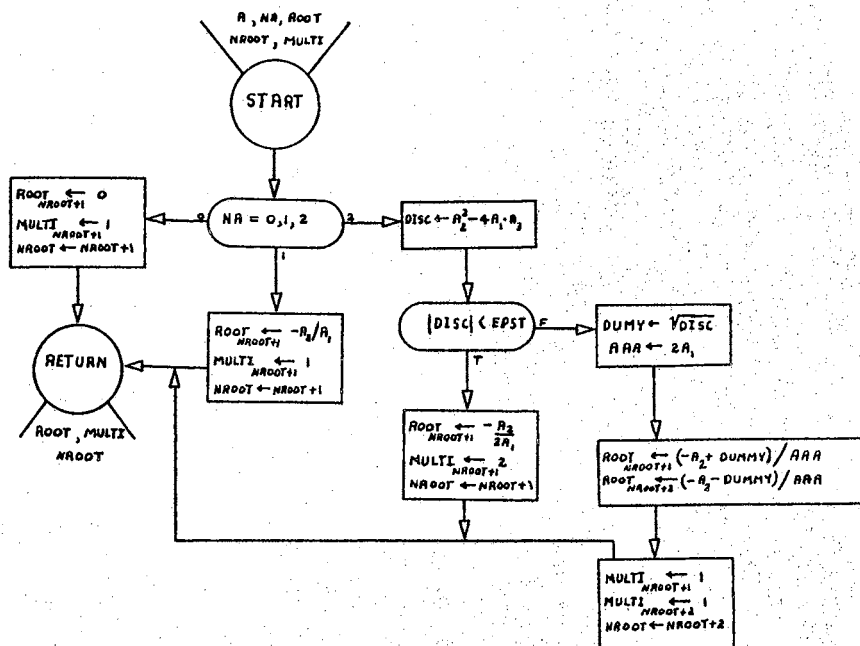


Figure 13.2. (Continued)

QUAD



COMSQT

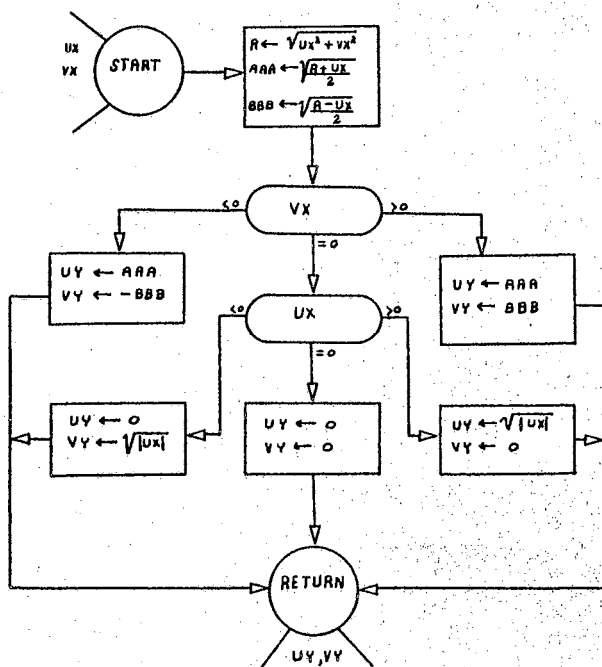


Figure 13.2. (Continued)

TABLE XXI-A

SINGLE PRECISION PROGRAM FOR G.C.D. - MULLER'S METHOD

\$JOB 10414

```

C *****
C *
C * SINGLE PRECISION PROGRAM FOR G.C.D. - MULLER'S METHOD
C *
C *
C * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
C * POLYNOMIAL OF MAXIMUM DEGREE 25. ALL MULTIPLE ROOTS ARE REMOVED BY
C * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL
C * AND ITS DERIVATIVE. THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED
C * AND THEIR MULTIPLICITIES DETERMINED.
C *
C *****
1  COMPLEX P,DIAPP,ROOT,DP,RK,H
2  DIMENSION P(26),DIAPP(25),ROOT(25),MULTI(25),DP(26),RK(26)
3  DIMENSION H(26),ANAME(2)
4  COMMON DUM1,EPSQ,EPSCNV,IO2,MAX
5  DATA PNAME,HNAME,ONAME/2HP(,2HH(,2HD(/
6  DATA ANAME(1),ANAME(2)/4HMULL,4HERS /
7  IO1=5
8  IO2=6
9  10 NROOT=0
10 READ(IO1,1000) NOPOLY, NP, NIAP, MAX, EPSG, EPSCNV, EPSQ, EPSMUL, XSTART, X
    IEND, KCHECK
11 IF(KCHECK.EQ.1) STOP
12 KKK=NP+1
13 READ(IO1,1010) (P(I),I=1,KKK)
14 IF(NIAP.EQ.0) GO TO 20
15 READ(IO1,1020) (DIAPP(I),I=1,NIAP)
16 20 KKK=NP+1
17 WRITE(IO2,1030) ANAME(1),ANAME(2),NOPOLY, NP
18 WRITE(IO2,1035) (PNAME,I,P(I),I=1,KKK)
19 WRITE(IO2,2060)
20 WRITE(IO2,2000) NIAP
21 WRITE(IO2,2010) MAX
22 WRITE(IO2,2070) EPSG
23 WRITE(IO2,2020) EPSCNV
24 WRITE(IO2,2030) EPSMUL
25 WRITE(IO2,2040) XSTART
26 WRITE(IO2,2050) XEND
27 IF(NP.GT.2) GO TO 90
28 CALL QUAD(P,NP,ROOT,NROOT,MULTI,EPSCNV)
29 85 WRITE(IO2,1037)
30 WRITE(IO2,1086) (I,ROOT(I),MULTI(I),I=1,NROOT)
31 GO TO 10
32 90 CALL DERIV(P,NP,DP,NDP)
33 CALL GCD(P,NP,DP,NDP,RK,NRK,EPSCNV)
34 CALL DIVIDE(P,NP,RK,NRK,H,NH)
35 KKK=NH+1
36 WRITE(IO2,1060)
37 WRITE(IO2,1035) (HNAME,I,H(I),I=1,KKK)
38 IF(NH.GT.2) GO TO 150
39 CALL QUAD(H,NH,ROOT,NROOT,MULTI,EPSCNV)
40 CALL MULTIPI(P,NP,ROOT,NROOT,MULTI,EPSCNV,NOPOLY)
41 GO TO 85
42 150 CALL MULLER(H,NH,DIAPP,NIAP,XSTART,XEND,EPSCNV,P,NP,NOPOLY)
43 GO TO 10
44 1086 FORMAT(2X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,11X,23HS
    IOLVED BY DIRECT METHOD)

```

TABLE XXI-A (Continued)

```

45 1000 FORMAT(3(I2,1X),9X,I3,1X,4(E6.0,1X),13X,2(E7.0,1X),I1)
46 1010 FORMAT(2E30.0)
47 1020 FORMAT(2E30.0)
48 1030 FORMAT(1H1,10X,41HGREATEST COMMON DIVISOR METHOD USED WITH ,2(A4),
      135HMETHOD TO FIND ZEROS OF POLYNOMIALS/11X,18HPOLYNOMIAL NUMBER ,I
      22,11H OF DEGREE ,I2///1X,28HTHE COEFFICIENTS OF P(X) ARE//)
49 1035 FORMAT(3X,A2,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
50 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,I2)
51 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
52 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,E9.2)
53 2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,E9.2)
54 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,E9.2)
55 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,E9.2)
56 2060 FORMAT(//1X)
57 2070 FORMAT(1X,31HTEST FOR ZERO IN SUBROUTINE GCD,3X,E9.2)
58 1037 FORMAT(///,1X,13HZEROS OF P(X),37X,14HMULTIPLICITIES//)
59 1060 FORMAT(//1X,42HTHE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE//)
60 END

```

```

61 SUBROUTINE DERIV(A,NA,B,NB)

```

```

C *****
C *
C * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF *
C * ITS DERIVATIVE P'(X), *
C *
C *****

```

```

62 COMPLEX A,B
63 DIMENSION A(26),B(26)
64 IF(NA.GE.2) GO TO 30
65 IF(NA.EQ.1) GO TO 20
66 B(1)=0.0
67 NB=-1
68 GO TO 50
69 20 B(1)=A(1)
70 NB=0
71 GO TO 50
72 30 NB=NA-1
73 DO 40 I=1,NA
74 BBB=NA-I+1
75 40 B(I)=BBB*A(I)
76 50 RETURN
77 END

```


TABLE XXI-A (Continued)

```

78      SUBROUTINE DIVIDE(F,NF,G,NG,H,NH)
C      *****
C      *
C      * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE
C      * QUOTIENT POLYNOMIAL H(X) = F(X)/G(X).
C      *
C      *****
79      COMPLEX F,G,H,X,PX,TERM
80      DIMENSION F(26),G(26),H(26)
81      COMMON DUM1,DUM2,EPS,IO2,MAX
82      IF(NG.GE.2) GO TO 60
83      IF(NG.EQ.1) GO TO 30
84      NH=NF
85      KKK=NF+1
86      DO 20 I=1,KKK
87      20 H(I)=F(I)/G(1)
88      GO TO 100
89      30 X=-G(2)/G(1)
90      NNF=NF
91      CALL HORNER(NNF,F,X,H,PX)
92      NH=NNF-1
93      GO TO 100
94      60 NH=NF-NG
95      H(1)=F(1)/G(1)
96      KKK=NH+1
97      DO 90 I=2,KKK
98      TERM=F(I)
99      J=2
100     K=I-1
101     70 TERM=TERM-G(J)*H(K)
102     IF(J.EQ.1) GO TO 90
103     IF(J.GE.NG+1) GO TO 90
104     J=J+1
105     K=K-1
106     GO TO 70
107     90 H(I)=TERM/G(1)
108     100 RETURN
109     END

```

TABLE XXI-A (Continued)

```

110      SUBROUTINE GCD(P,NP,DP,NDP,RK,NK,EPSP)
C      *****
C      *
C      * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG.
C      * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND
C      * DP(X).
C      *
C      *****
111      COMPLEX R1,R2,RK,U,T,P,DP
112      DIMENSION R1(26),R2(26),T(26),RK(26),P(26),DP(26)
113      COMMON DUM1,DUM2,EPS,I02,MAX
114      N1=NP
115      N2=NDP
116      KKK=N1+1
117      DO 10 I=1,KKK
118      10 R1(I)=P(I)
119      KKK=N2+1
120      DO 20 I=1,KKK
121      20 R2(I)=DP(I)
122      50 U=R1(1)/R2(1)
123      KKK=N2+1
124      DO 70 I=1,KKK
125      70 T(I)=R1(I)-U*R2(I)
126      T(1)=0.0
127      IF(N1.EQ.N2) GO TO 90
128      KKK=N1+1
129      NNN=N2+2
130      DO 80 I=NNN,KKK
131      80 T(I)=R1(I)
132      90 KKK=N1+1
133      DO 100 I=1,KKK
134      BBB=CABS(T(I))
135      IF(BBB.GT.EPSP) GO TO 120
136      100 CONTINUE
137      NK=N2
138      KKK=NK+1
139      DO 110 I=2,KKK
140      110 RK(I)=R2(I)/R2(1)
141      RK(1)=1.0
142      GO TO 200
143      120 M=N1+1-I
144      IF(M.LT.N2) GO TO 160
145      N1=M
146      KKK=N1+1
147      NNN=I-1
148      DO 150 J=1,KKK
149      150 R1(J)=T(NNN+J)
150      GO TO 50
151      160 N1=N2
152      KKK=N1+1
153      DO 170 J=1,KKK
154      170 R1(J)=R2(J)
155      N2=M
156      KKK=N2+1
157      NNN=I-1
158      DO 180 J=1,KKK
159      180 R2(J)=T(NNN+J)
160      GO TO 50
161      200 RETURN
162      END

```

TABLE XXI-A (Continued)

```

163     SUBROUTINE MULTIP(A,NA,ROOT,NR,MULTI,EPSPX)
C     *****
C     *
C     * GIVEN NR ZEROS OF A POLYNOMIAL, SUBROUTINE MULTIP COMPUTES THEIR
C     * MULTIPLICITIES.
C     *
C     *****
164     COMPLEX A,ROOT,WORK,PX,DPX,B
165     DIMENSION A(26),ROOT(25),WORK(26),MULTI(25),B(26)
166     COMMON DUM1,DUM2,EPS,IO2,MAX
167     DO 50 I=1,NR
168     MULTI(I)=0
169     KKK=NA+1
170     DO 20 J=1,KKK
171     20 WORK(J)=A(J)
172     NWORK=NA
173     25 CALL HORNER(NWORK,WORK,ROOT(I),B,PX)
174     NWORK=NWORK-1
175     BBB=CABS(PX)
176     IF(BBB.GT.EPSPX) GO TO 45
177     MULTI(I)=MULTI(I)+1
178     IF(NWORK.LT.1) GO TO 50
179     KKK=NWORK+1
180     DO 40 J=1,KKK
181     40 WORK(J)=B(J)
182     GO TO 25
183     45 IF(MULTI(I).EQ.0) WRITE(IO2,1000) EPSPX,I,ROOT(I)
184     50 CONTINUE
185     100 RETURN
186     1000 FORMAT(///14H THE EPSILON (,E14.7,49H) CHECK IN SUBROUTINE MULTIP
187     1INDICATES THAT ROOT(I,12,4H) = ,E14.7,3H + ,E14.7,2H I/80H IS NOT C
2LOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIPLIC
3ITY 0//)
END

```

TABLE XXI-A (Continued)

```

188      SUBROUTINE MULLER(A,NP,OIAPP,NAPP,XSTART,XEND,EPSM,P,NDEG,NOPOLY)
C      *****
C      *
C      * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
C      * POLYNOMIAL OF MAXIMUM DEGREE 25. THROUGH THREE GIVEN POINTS THE
C      * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC. THE ZERO OF THE QUADRATIC
C      * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.
C      * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.
C      *
C      *****
189      COMPLEX PX3,PX2,ROOT,X1,APP,X2,WORK,X3,B,X4,A,PX1,RAPP,PX4,H3,Q4
190      COMPLEX OIAPP,P
191      DIMENSION ROOT(25),MULT(25),APP(25,3),WORK(26),B(26),A(26),RAPP(25
1,3)
192      DIMENSION OIAPP(25),P(26)
193      DATA PNAME,DNAME/2HP(,2HD(/
194      LOGICAL CONV
195      COMMON EPSRT,EPSD,EPS,I02,MAX
196      EPSRT=0.999
197      NROOT=0
198      IROOT=0
199      IPATH=1
200      NALTER=0
201      ITIME=0
202      NOMULT=0
203      IAPP=1
204      ITER=1
205      IF(NAPP.NE.0) GO TO 18
206      NAPP=NP
207      CALL GENAPP(APP,NAPP,XSTART)
208      GO TO 27
209      18 DO 20 I=1,NAPP
210      20 APP(I,2)=OIAPP(I)
211      DO 25 I=1,NAPP
212      APP(I,1)=0.9*APP(I,2)
213      25 APP(I,3)=1.1*APP(I,2)
214      27 KKK=NP+1
215      DO 30 I=1,KKK
216      30 WORK(I)=A(I)
217      NWORK=NP
218      40 X1=APP(IAPP,1)
219      X2=APP(IAPP,2)
220      X3=APP(IAPP,3)
221      CALL HORNER(NWORK,WORK,X1,B,PX1)
222      CALL HORNER(NWORK,WORK,X2,B,PX2)
223      CALL HORNER(NWORK,WORK,X3,B,PX3)
224      50 CALL CALC(X1,X2,X3,PX1,PX2,PX3,X4,Q4,H3)
225      55 CALL HORNER(NWORK,WORK,X4,B,PX4)
226      AB1=CABS(PX3)
227      IF(AB1.EQ.0.0) GO TO 60
228      QQQ=CABS(PX4)/CABS(PX3)
229      IF(QQQ.LE.10.) GO TO 60
230      Q4=0.5*Q4
231      X4=X3+H3*Q4
232      GO TO 55
233      60 CALL TEST(X3,X4,CONV)
234      IF(CONV) GO TO 120
235      IF(ITER.LT.MAX) GO TO 110
236      CALL ALTER(APP(IAPP,1),APP(IAPP,2),APP(IAPP,3),NALTER,ITIME)
237      IF(NALTER.GT.5) GO TO 75

```

TABLE XXI-A (Continued)

```

238     ITER=1
239     GO TO 40
240     75 IF(IAPP.LT.NAPP) GO TO 100
241     IF(XEND.EQ.0.0) GO TO 70
242     IF(XSTART.GT.XEND) GO TO 70
243     NAPP=NP
244     CALL GENAPP(APP,NAPP,XSTART)
245     IAPP=0
246     GO TO 100
247     70 WRITE(IO2,1090)
248     KKK=NWORK+1
249     WRITE(IO2,1035) (DNAME,J,WORK(J),J=1,KKK)
250     80 IF(NROOT.EQ.0) GO TO 90
251     WRITE(IO2,1060)
252     IF(IPATH.EQ.1) GO TO 82
253     81 IPATH=2
254     CALL BETTER(A,NP,ROOT,NROOT,RAPP,IROOT,MULT)
255     WRITE(IO2,1200)
256     CALL MULTI(P,NDEG,ROOT,NROOT,MULT,EPSM)
257     82 IF(NROOT.EQ.0) GO TO 90
258     IF(IROOT.EQ.0) GO TO 85
259     WRITE(IO2,1080)
260     DO 83 I=1,IROOT
261     83 WRITE(IO2,1085) I,ROOT(I),MULT(I),RAPP(I,2)
262     IF(IROOT.LT.NROOT) GO TO 85
263     GO TO 87
264     85 KKK=IROOT+1
265     WRITE(IO2,1086) (I,ROOT(I),MULT(I),I=KKK,NROOT)
266     87 IF(IPATH.EQ.1) GO TO 81
267     RETURN
268     90 WRITE(IO2,1070) NOPOLY
269     RETURN
270     100 IAPP=IAPP+1
271     ITER=1
272     NALTER=0
273     GO TO 40
274     120 NROOT=NROOT+1
275     IROOT=NROOT
276     MULT(NROOT)=1
277     NOMULT=NOMULT+1
278     ROOT(NROOT)=X4
279     RAPP(NROOT,1)=APP(IAPP,1)
280     RAPP(NROOT,2)=APP(IAPP,2)
281     RAPP(NROOT,3)=APP(IAPP,3)
282     125 IF(NOMULT.LT.NP) GO TO 130
283     GO TO 80
284     130 CALL HORNER(NWORK,WORK,X4,B,PX4)
285     NWORK=NWORK-1
286     KKK=NWORK+1
287     DO 140 I=1,KKK
288     140 WORK(I)=B(I)
289     CALL HORNER(NWORK,WORK,X4,B,PX4)
290     CCC=CABS(PX4)
291     IF(CCC.LT.EPSM) GO TO 150
292     IF(NWORK.GT.2) GO TO 75
293     IROOT=NROOT
294     CALL QUAD(WORK,NWORK,ROOT,NROOT,MULT,EPS0)
295     GO TO 80
296     150 MULT(NROOT)=MULT(NROOT)+1
297     NOMULT=NOMULT+1

```

TABLE XXI-A (Continued)

```

298      GO TO 125
299      110 X1=X2
300      X2=X3
301      X3=X4
302      PX1=PX2
303      PX2=PX3
304      PX3=PX4
305      ITER=ITER+1
306      GO TO 50
307      1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
          1ZEROS WERE FOUND//)
308      1080 FORMAT(///1X,13HROOTS OF P(X),37X,14HMULTIPLICITIES,11X,21HINITIAL
          1 APPROXIMATION//)
309      1070 FORMAT(///1X,42HNO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
310      1035 FORMAT(3X,A2,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
311      1085 FORMAT(2X,5HROOT(I,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,10X,E14.
          17,3H + ,E14.7,2H I)
312      1086 FORMAT(2X,5HROOT(I,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,11X,23HS
          1OLVED BY DIRECT METHOD)
313      1060 FORMAT(///35H BEFORE ATTEMPT TO IMPROVE ACCURACY)
314      1200 FORMAT(///1X,37HAFTER THE ATTEMPT TO IMPROVE ACCURACY)
315      END

```

TABLE XXI-A (Continued)

```

316 SUBROUTINE BETTER(A,NP,ROOT,NROOT,RAPP,IROOT,MULT)
C *****
C *
C * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND *
C * BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO *
C * THE FULL, UNDEFLATED POLYNOMIAL. *
C *
C *****
317 COMPLEX ROOT,A,BAPP,X1,X2,X3,PX1,PX2,PX3,B,ROOTS,X4,RAPP,Q4,H3
318 LOGICAL CONV
319 DIMENSION ROOT(25),A(26),BAPP(25,3),B(26),ROOTS(25),RAPP(25,3),MUL
IT(25)
320 COMMON EPSRT,EPSO,EPS,IO2,MAX
321 IF(NROOT.LE.1) RETURN
322 L=0
323 DO 10 I=1,NROOT
324 BAPP(I,1)=ROOT(I)*EPSRT
325 BAPP(I,2)=ROOT(I)
326 10 BAPP(I,3)=ROOT(I)*(2.0-EPSRT)
327 DO 100 J=1,NROOT
328 X1=BAPP(J,1)
329 X2=BAPP(J,2)
330 X3=BAPP(J,3)
331 ITER=1
332 CALL HORNER(NP,A,X1,B,PX1)
333 CALL HORNER(NP,A,X2,B,PX2)
334 20 CALL HORNER(NP,A,X3,B,PX3)
335 CALL CALC(X1,X2,X3,PX1,PX2,PX3,X4,Q4,H3)
336 30 CALL TEST(X3,X4,CONV)
337 IF(CONV) GO TO 50
338 IF(ITER.LT.MAX) GO TO 40
339 WRITE(IO2,1000) J,ROOT(J),MAX
340 WRITE(IO2,1010) X4
341 IF(J.LT.IROOT) GO TO 33
342 IF(J.EQ.IROOT) GO TO 35
343 GO TO 100
344 33 KKK=IROOT-1
345 DO 34 K=J,KKK
346 RAPP(K,1)=RAPP(K+1,1)
347 RAPP(K,2)=RAPP(K+1,2)
348 34 RAPP(K,3)=RAPP(K+1,3)
349 35 IROOT=IROOT-1
350 GO TO 100
351 40 X1=X2
352 X2=X3
353 X3=X4
354 PX1=PX2
355 PX2=PX3
356 ITER=ITER+1
357 GO TO 20
358 50 L=L+1
359 ROOTS(L)=X4
360 MULT(L)=MULT(J)
361 100 CONTINUE
362 IF(L.EQ.0) GO TO 120
363 DO 110 I=1,L
364 110 ROOT(I)=ROOTS(I)
365 NROOT=L
366 RETURN
367 120 NROOT=0

```

TABLE XXI-A (Continued)

```

368     RETURN
369     1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,
      1E14.7,3H + ,E14.7,2H 1,24H DID NOT CONVERGE AFTER ,I3,11H ITERATIO
      2NS)
370     1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,E14.7,3H + ,E14.7,2H I//)
371     END

```

```

372     SUBROUTINE CALC(X1,X2,X3,PX1,PX2,PX3,Q4,H3)
C     *****
C     *
C     * GIVEN THREE APPROXIMATIONS X(N-2), X(N-1), AND X(N), SUBROUTINE CALC
C     * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF
C     * THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION
C     * X(N+1) TO THE ZERO OF THE POLYNOMIAL.
C     *
C     *****
373     COMPLEX PX3,PX2,X1,X2,X3,PX1,H3,H2,Q3,D,B,C,DISC,DEN1,DEN2,Q4,X4
374     COMPLEX TEST,CCC
375     COMPLEX CMLX
376     COMMON EPSRT,EPSO,EPS,IO2,MAX
377     H3=X3-X2
378     H2=X2-X1
379     Q3=H3/H2
380     D=Q3*(PX3-((1.0+Q3)*PX2)+(Q3*PX1))
381     B=(((2.0*Q3)+1.0)*PX3)-((1.0+Q3)*(1.0+Q3)*PX2)+(Q3*Q3*PX1)
382     C=(1.0+Q3)*PX3
383     DISC=(B*B)-(4.0*D*C)
384     AAA=CABS(DISC)
385     IF(AAA.EQ.0.0) GO TO 5
386     GO TO 7
387     5 THETA=0.0
388     GO TO 9
389     7 DISCI=AIMAG(DISC)
390     DISCR=REAL(DISC)
391     THETA=ATAN2(DISCI,DISCR)
392     9 RAD=SQRT(AAA)
393     ANGLE=THETA/2.0
394     CCC=CMLX(COS(ANGLE),SIN(ANGLE))
395     TEST=RAD*CCC
396     DEN1=B+TEST
397     DEN2=B-TEST
398     AAA=CABS(DEN1)
399     BBB=CABS(DEN2)
400     IF(AAA.LT.BBB) GO TO 10
401     IF(AAA.EQ.0.0) GO TO 60
402     Q4=(-2.0*C)/DEN1
403     GO TO 50
404     10 IF(BBB.EQ.0.0) GO TO 60
405     Q4=(-2.0*C)/DEN2
406     GO TO 50
407     50 X4=X3+(H3*Q4)
408     RETURN
409     60 Q4=(1.0,0.0)
410     GO TO 50
411     END

```


TABLE XXI-A (Continued)

```

412     SUBROUTINE GENAPP(APP,NAPP,XSTART)
C     *****
C     *
C     * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE *
C     * DEGREE OF THE ORIGINAL POLYNOMIAL. *
C     *
C     *****
413     COMPLEX APP
414     COMPLEX CMLX
415     DIMENSION APP(25,3)
416     COMMON EPS1, EPS2, EPS3, IO2, MAX
417     IF(XSTART.EQ.0.0) XSTART=0.5
418     BETA=0.2617994
419     DO 10 I=1,NAPP
420     U=XSTART*COS(BETA)
421     V=XSTART*SIN(BETA)
422     APP(I,2)=CMLX(U,V)
423     BETA=BETA+0.5235988
424     10 XSTART=XSTART+0.5
425     DO 20 I=1,NAPP
426     APP(I,1)=0.9*APP(I,2)
427     20 APP(I,3)=1.1*APP(I,2)
428     RETURN
429     END

```

TABLE XXI-A (Continued)

```

430      SUBROUTINE ALTER(X1,X2,X3,NALTER,ITIME)
C      *****
C      *
C      * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO
C      * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.
C      *
C      *****
431      COMPLEX X1,X2,X3
432      COMPLEX CMPLX
433      COMMON EPS1, EPS2, EPS3, IO2, MAX
434      IF(ITIME.NE.0) GO TO 5
435      ITIME=1
436      WRITE(IO2,1010) MAX
437      5 IF(NALTER.EQ.0) GO TO 10
438      WRITE(IO2,1000) X1,X2,X3
439      GO TO 20
440      10 Y=AIMAG(X2)
441      X=REAL(X2)
442      R=CABS(X2)
443      BETA=ATAN2(Y,X)
444      WRITE(IO2,1020) X1,X2,X3
445      20 NALTER=NALTER+1
446      IF(NALTER.GT.5) RETURN
447      GO TO (30,40,30,40,30),NALTER
448      30 X2=-X2
449      GO TO 50
450      40 BETA=BETA+1.0471976
451      X2R=R*COS(BETA)
452      X2I=R*SIN(BETA)
453      X2=CMPLX(X2R,X2I)
454      50 X1=0.9*X2
455      X3=1.1*X2
456      RETURN
457      1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
1TER ,I3,12H ITERATIONS.//)
458      1000 FORMAT(1X,5HX1 = ,E14.7,3H + ,E14.7,2H I,10X,22HALTERED APPROXIMAT
1IONS/1X,5HX2 = ,E14.7,3H + ,E14.7,2H I/1X,5HX3 = ,E14.7,3H + ,E14.
27,2H I/)
459      1020 FORMAT(1H0,5HX1 = ,E14.7,3H + ,E14.7,2H I,10X,22HINITIAL APPROXIMA
1TIONS/1X,5HX2 = ,E14.7,3H + ,E14.7,2H I/1X,5HX3 = ,E14.7,3H + ,E14
2.7,2H I/)
460      END

```

TABLE XXI-A (Continued)

```

461      SUBROUTINE TEST(X3,X4,CONV)
C      *****
C      *
C      * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
C      * IMATIONS BY TESTING THE EXPRESSION
C      * ABSOLUTE VALUE OF (X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).
C      * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
C      *
C      *****
462      COMPLEX X3,X4
463      LOGICAL CONV
464      COMMON EPSRT,EPSO,EPS,IO2,MAX
465      AAA=CABS(X4-X3)
466      DENOM=CABS(X4)
467      IF(DENOM.LT.EPSO) GO TO 20
468      IF(AAA/DENOM.LT.EPS) GO TO 10
469      5 CONV=.FALSE.
470      GO TO 100
471      10 CONV=.TRUE.
472      GO TO 100
473      20 IF(AAA.LT.EPSO) GO TO 10
474      GO TO 5
475      100 RETURN
476      END

477      SUBROUTINE HORNER(NA,A,X,B,PX)
C      *****
C      *
C      * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D.
C      * SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE
C      * FACTOR (X-D).
C      *
C      *****
478      COMPLEX X,PX,B,A
479      DIMENSION A(26),B(26)
480      B(1)=A(1)
481      NUM=NA+1
482      DO 10 I=2,NUM
483      10 B(I)=A(I)+B(I-1)*X
484      PX=B(NUM)
485      RETURN
486      END

```

TABLE XXI-A (Continued)

```

487      SUBROUTINE QUAD(A,NA,ROOT,NROOT,MULTI,EPST)
C      *****
C      *
C      * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
C      * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE *
C      * QUADRATIC IS DONE USING THE QUADRATIC FORMULA. *
C      *
C      *****
488      COMPLEX A,DISC,ROOT,DUMMY,AAA
489      COMPLEX CSQRT
490      DIMENSION A(26),ROOT(25),MULTI(25)
491      IF(NA.EQ.2) GO TO 7
492      IF(NA.EQ.1) GO TO 5
493      ROOT(NROOT+1)=0.0
494      MULTI(NROOT+1)=1
495      NROOT=NROOT+1
496      GO TO 50
497      5 ROOT(NROOT+1)=-A(2)/A(1)
498      MULTI(NROOT+1)=1
499      NROOT=NROOT+1
500      GO TO 50
501      7 DISC=A(2)*A(2)-(4.0*A(1)*A(3))
502      BBB=CABS(DISC)
503      IF(BBB.LT.EPST) GO TO 10
504      DUMMY=CSQRT(DISC)
505      AAA=2.0*A(1)
506      ROOT(NROOT+1)=(-A(2)+DUMMY)/AAA
507      ROOT(NROOT+2)=(-A(2)-DUMMY)/AAA
508      MULTI(NROOT+1)=1
509      MULTI(NROOT+2)=1
510      NROOT=NROOT+2
511      GO TO 50
512      10 ROOT(NROOT+1)=[-A(2)]/(2.0*A(1))
513      MULTI(NROOT+1)=2
514      NROOT=NROOT+1
515      50 RETURN
516      END

```

TABLE XXI-B

DOUBLE PRECISION PROGRAM FOR G.C.D. - MULLER'S METHOD

```

$JOB 10414
C *****
C *
C * DOUBLE PRECISION PROGRAM FOR G.C.D. - MULLER'S METHOD
C *
C *
C * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
C * POLYNOMIAL OF MAXIMUM DEGREE 25. ALL MULTIPLE ROOTS ARE REMOVED BY
C * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL
C * AND ITS DERIVATIVE. THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED
C * AND THEIR MULTIPLICITIES DETERMINED.
C *
C *****
1  DOUBLE PRECISION UP,VP,UOIAPP,VOIAPP,UROOT,VROOT,UDP,VDP,URK,VRK
2  DOUBLE PRECISION UH,VH,EPCHK,UDUMMY,VDUMMY,EPSQ,EPSMUL,EPSCG,EPSCN
3  DOUBLE PRECISION DUM1
4  DIMENSION ANAME(2),UP(26),VP(26),UOIAPP(25),VOIAPP(25),UROOT(25),V
5  IROOT(25),UDP(26),VDP(26),URK(26),VRK(26),UH(26),VH(26),MULTI(25)
6  COMMON DUM1,EPSCG,EPSCNV,IO2,MAX
7  DATA PNAME,HNAME,DNAME/2HP(,2HH(,2HD(/
8  DATA ANAME(1),ANAME(2)/4HMULL,4HERS /
9  IO1=5
10 IO2=6
11 10 NROOT=0
12 READ(IO1,1000) NOPOLY,NP,NIAP,MAX,EPSCG,EPSCNV,EPSCQ,EPSCMUL,XSTART,X
13 IEND,KCHECK
14 IF(KCHECK.EQ.1) STOP
15 KKK=NP+1
16 READ(IO1,1010) (UP(I),VP(I),I=1,KKK)
17 IF(NIAP.EQ.0) GO TO 20
18 READ(IO1,1020) (UOIAPP(I),VOIAPP(I),I=1,NIAP)
19 20 KKK=NP+1
20 WRITE(IO2,1030) ANAME(1),ANAME(2),NOPOLY,NP
21 WRITE(IO2,1035) (PNAME,I,UP(I),VP(I),I=1,KKK)
22 WRITE(IO2,2060)
23 WRITE(IO2,2000) NIAP
24 WRITE(IO2,2010) MAX
25 WRITE(IO2,2070) EPSCG
26 WRITE(IO2,2020) EPSCNV
27 WRITE(IO2,2030) EPSCMUL
28 WRITE(IO2,2040) XSTART
29 WRITE(IO2,2050) XEND
30 IF(NP.GT.2) GO TO 90
31 CALL QUAD(UP,VP,NP,UROOT,VROOT,MULTI,EPSCQ)
32 85 WRITE(IO2,1037)
33 WRITE(IO2,1086) (I,UROOT(I),VROOT(I),MULTI(I),I=1,NROOT)
34 GO TO 10
35 90 CALL DERIV(UP,VP,NP,UDP,VDP,NDP)
36 CALL GCD(UP,VP,NP,UDP,VDP,NDP,URK,VRK,NRK,EPSCG)
37 CALL DIVIDE(UP,VP,NP,URK,VRK,NRK,UH,VH,NH)
38 KKK=NH+1
39 WRITE(IO2,1060)
40 WRITE(IO2,1035) (HNAME,I,UH(I),VH(I),I=1,KKK)
41 IF(NH.GT.2) GO TO 150
42 CALL QUAD(UH,VH,NH,UROOT,VROOT,NROOT,MULTI,EPSCQ)
43 CALL MULTI(UP,VP,NP,UROOT,VROOT,NROOT,MULTI,EPSCMUL)
44 GO TO 85
45 150 CALL MULLER(UH,VH,NH,UOIAPP,VOIAPP,NIAP,XSTART,XEND,EPSCMUL,UP,VP,N

```

TABLE XXI-B (Continued)

```

      IP,NOPOLY)
44      GO TO 10
45      1000 FORMAT(3(I2,1X),9X,I3,1X,4(D6.0,1X),13X,2(D7.0,1X),11)
46      1010 FORMAT(2D30.0)
47      1020 FORMAT(2D30.0)
48      1030 FORMAT(1H1,10X,41HGREATEST COMMON DIVISOR METHOD USED WITH ,2(A4),
      135METHOD TO FIND ZEROS OF POLYNOMIALS/11X,18HPOLYNOMIAL NUMBER ,I
      22,11H OF DEGREE ,I2///1X,28HTHE COEFFICIENTS OF P(X) ARE//)
49      1035 FORMAT(3X,A2,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
50      1037 FORMAT(///1X,13HZEROS OF P(X),55X,14HMULTIPLICITIES//)
51      1086 FORMAT(3X,5HROOT(I,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,8X,23HS
      10LVED BY DIRECT METHOD)
52      1060 FORMAT(///1X,42HTHE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE//)
53      2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,I2)
54      2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
55      2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
56      2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)
57      2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
58      2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
59      2060 FORMAT(//1X)
60      2070 FORMAT(1X,31HTEST FOR ZERO IN SUBROUTINE GCD,3X,D9.2)
61      END

62      SUBROUTINE DERIV(UA,VA,NA,UB,VB,NB)
      C *****
      C *
      C * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF *
      C * ITS DERIVATIVE P'(X). *
      C *
      C *****
63      DOUBLE PRECISION UA,VA,UB,VB
64      DIMENSION UA(26),VA(26),UB(26),VB(26)
65      IF(NA.GE.2) GO TO 30
66      IF(NA.EQ.1) GO TO 20
67      UB(1)=0.0
68      VB(1)=0.0
69      NB=-1
70      GO TO 50
71      20 UB(1)=UA(1)
72      VB(1)=VA(1)
73      NB=0
74      GO TO 50
75      30 NB=NA-1
76      DO 40 I=1,NA
77      BBB=NA-I+1
78      UB(I)=BBB*UA(I)
79      40 VB(I)=BBB*VA(I)
80      50 RETURN
81      END

```

TABLE XXI-B (Continued)

```

82      SUBROUTINE DIVIDE(UF,VF,NF,UG,VG,NG,UH,VH,NH)
C      *****
C      *
C      * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE
C      * QUOTIENT POLYNOMIAL H(X) = F(X)/G(X).
C      *
C      *****
83      DOUBLE PRECISION UF,VF,UG,VG,UH,VH,UX,VX,UPX,VPX,UTERM,VTERM,UDUMM
1Y,VDUMMY,EPS
84      DOUBLE PRECISION DUM1,DUM2
85      DOUBLE PRECISION DENOM
86      DIMENSION UF(26),VF(26),UG(26),VG(26),UH(26),VH(26)
87      COMMON DUM1,DUM2,EPS,IO2,MAX
88      IF(NG.GE.2) GO TO 60
89      IF(NG.EQ.1) GO TO 30
90      NH=NF
91      KKK=NF+1
92      DO 20 I=1,KKK
93      DENOM=UG(I)*UG(I)+VG(I)*VG(I)
94      UH(I)=(UF(I)*UG(I)+VF(I)*VG(I))/DENOM
95      20  VH(I)=(VF(I)*UG(I)-UF(I)*VG(I))/DENOM
96      GO TO 100
97      30  UDUMMY=-1.0*UG(2)
98      VDUMMY=-1.0*VG(2)
99      DENOM=UG(1)*UG(1)+VG(1)*VG(1)
100     UX=(UDUMMY*UG(1)+VDUMMY*VG(1))/DENOM
101     VX=(VDUMMY*UG(1)-UDUMMY*VG(1))/DENOM
102     NNF=NF
103     CALL HORNER(NNF,UF,VF,UX,VX,UH,VH,UPX,VPX)
104     NH=NNF-1
105     GO TO 100
106     60  NH=NF-NG
107     DENOM=UG(1)*UG(1)+VG(1)*VG(1)
108     UH(1)=(UF(1)*UG(1)+VF(1)*VG(1))/DENOM
109     VH(1)=(VF(1)*UG(1)-UF(1)*VG(1))/DENOM
110     KKK=NH+1
111     DO 95 I=2,KKK
112     UTERM=UF(I)
113     VTERM=VF(I)
114     J=2
115     K=I-1
116     70  UTERM=UTERM-(UG(J)*UH(K)-VG(J)*VH(K))
117     VTERM=VTERM-(VG(J)*UH(K)+UG(J)*VH(K))
118     IF(J.EQ.I) GO TO 90
119     IF(J.GE.NG+1) GO TO 90
120     J=J+1
121     K=K-1
122     GO TO 70
123     90  DENOM=UG(1)*UG(1)+VG(1)*VG(1)
124     UH(I)=(UTERM*UG(1)+VTERM*VG(1))/DENOM
125     95  VH(I)=(VTERM*UG(1)-UTERM*VG(1))/DENOM
126     100 RETURN
127     END

```

TABLE XXI-B (Continued)

```

128      SUBROUTINE QUAD(UA,VA,NA,UROOT,VROOT,NROOT,MULTI,EPST)
C      *****
C      *
C      * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
C      * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE *
C      * QUADRATIC IS DONE USING THE QUADRATIC FORMULA. *
C      *
C      *****
129      DOUBLE PRECISION UA,VA,UROOT,VROOT,BBB,UAAA,VAAA,UDISC,VDISC,UDUMM
130      1Y,VDUMMY,RDUMMY,SDUMMY,EPST,UBBB,VBBB
131      DOUBLE PRECISION DSQRT
132      DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),MULTI(25)
133      IF(NA.EQ.2) GO TO 7
134      IF(NA.EQ.1) GO TO 5
135      UROOT(NROOT+1)=0.0
136      VROOT(NROOT+1)=0.0
137      MULTI(NROOT+1)=1
138      NROOT=NROOT+1
139      GO TO 50
140      5 BBB=UA(1)*UA(1)+VA(1)*VA(1)
141      UROOT(NROOT+1)=(-UA(2)*UA(1)-VA(2)*VA(1))/BBB
142      VROOT(NROOT+1)=(-VA(2)*UA(1)+UA(2)*VA(1))/BBB
143      MULTI(NROOT+1)=1
144      NROOT=NROOT+1
145      GO TO 50
146      7 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(1)*UA(3)-VA(1)*VA(3)))
147      VDISC=(VA(2)*UA(2)+UA(2)*VA(2))-(4.0*(VA(1)*UA(3)+UA(1)*VA(3)))
148      BBB=DSQRT(UDISC*UDISC+VDISC*VDISC)
149      IF(BBB.LT.EPST) GO TO 10
150      CALL COMSOT(UDISC,VDISC,UDUMMY,VDUMMY)
151      UBBB=-UA(2)+UDUMMY
152      VBBB=-VA(2)+VDUMMY
153      RDUMMY=-UA(2)-UDUMMY
154      SDUMMY=-VA(2)-VDUMMY
155      UAAA=2.0*UA(1)
156      VAAA=2.0*VA(1)
157      BBB=UAAA*UAAA+VAAA*VAAA
158      UROOT(NROOT+1)=(UBBB*UAAA+VBBB*VAAA)/BBB
159      VROOT(NROOT+1)=(VBBB*UAAA-UBBB*VAAA)/BBB
160      UROOT(NROOT+2)=(RDUMMY*UAAA+SDUMMY*VAAA)/BBB
161      VROOT(NROOT+2)=(SDUMMY*UAAA-RDUMMY*VAAA)/BBB
162      MULTI(NROOT+1)=1
163      MULTI(NROOT+2)=1
164      NROOT=NROOT+2
165      GO TO 50
166      10 UAAA=2.0*UA(1)
167      VAAA=2.0*VA(1)
168      BBB=UAAA*UAAA+VAAA*VAAA
169      UROOT(NROOT+1)=(-UA(2)*UAAA-VA(2)*VAAA)/BBB
170      VROOT(NROOT+1)=(-VA(2)*UAAA+UA(2)*VAAA)/BBB
171      MULTI(NROOT+1)=2
172      NROOT=NROOT+1
173      50 RETURN
      END

```


TABLE XXI-B (Continued)

```

174      SUBROUTINE GCD(UP,VP,NP,UDP,VDP,NDP,URK,VRK,NK,EPG)
C      *****
C      *
C      * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG.
C      * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND
C      * DP(X).
C      *
C      *****
175      DOUBLE PRECISION EPG,UP,VP,UDP,VDP,URK,VRK,UR1,VR1,UR2,VR2,UT,VT,
176      1UU,VU,BBB,EPS
177      DOUBLE PRECISION DUM1,DUM2
178      DOUBLE PRECISION DSQRT
179      DIMENSION UR1(26),VR1(26),UR2(26),VR2(26),UT(26),VT(26),URK(26),VR
180      1K(26),UP(26),VP(26),UDP(26),VDP(26)
181      COMMON DUM1,DUM2,EPS,IO2,MAX
182      N1=NP
183      N2=NDP
184      KKK=N1+1
185      DO 10 I=1,KKK
186      UR1(I)=UP(I)
187      10 VR1(I)=VP(I)
188      KKK=N2+1
189      DO 20 I=1,KKK
190      UR2(I)=UDP(I)
191      20 VR2(I)=VDP(I)
192      50 DENOM=UR2(1)*UR2(1)+VR2(1)*VR2(1)
193      UU=(UR1(1)*UR2(1)+VR1(1)*VR2(1))/DENOM
194      VU=(VR1(1)*UR2(1)-UR1(1)*VR2(1))/DENOM
195      KKK=N2+1
196      DO 70 I=1,KKK
197      UT(I)=UR1(I)-(UU*UR2(I)-VU*VR2(I))
198      70 VT(I)=VR1(I)-(VU*UR2(I)+UU*VR2(I))
199      UT(1)=0.0
200      VT(1)=0.0
201      IF(N1.EQ.N2) GO TO 90
202      KKK=N1+1
203      NNN=N2+2
204      DO 80 I=NNN,KKK
205      UT(I)=UR1(I)
206      80 VT(I)=VR1(I)
207      90 KKK=N1+1
208      DO 100 I=1,KKK
209      BBB=DSQRT(UT(I)*UT(I)+VT(I)*VT(I))
210      IF(BBB.GT.EPG) GO TO 120
211      100 CONTINUE
212      NK=N2
213      KKK=NK+1
214      DO 110 I=2,KKK
215      DENOM=UR2(1)*UR2(1)+VR2(1)*VR2(1)
216      URK(I)=(UR2(I)*UR2(1)+VR2(I)*VR2(1))/DENOM
217      110 VRK(I)=(VR2(I)*UR2(1)-UR2(I)*VR2(1))/DENOM
218      URK(1)=1.0
219      VRK(1)=0.0
220      GO TO 200
221      120 M=N1+1-I
222      IF(M.LT.N2) GO TO 160
223      N1=M
224      KKK=N1+1
225      NNN=I-1

```

TABLE XXI-B (Continued)

```

225      DO 150 J=1,KKK
226      UR1(J)=UT(NNN+J)
227      150 VR1(J)=VT(NNN+J)
228      GO TO 50
229      160 N1=N2
230      KKK=N1+1
231      DO 170 J=1,KKK
232      UR1(J)=UR2(J)
233      170 VR1(J)=VR2(J)
234      N2=M
235      KKK=N2+1
236      NNN=I-1
237      DO 180 J=1,KKK
238      UR2(J)=UT(NNN+J)
239      180 VR2(J)=VT(NNN+J)
240      GO TO 50
241      200 RETURN
242      END

243      SUBROUTINE MULTIP(UA,VA,NA,UROOT,VROOT,NR,MULTI,EPSPX)
C      *
C      *
C      * GIVEN NR ZEROS OF A POLYNOMIAL, SUBROUTINE MULTIP COMPUTES THEIR
C      * MULTIPLICITIES.
C      *
C      *
244      DOUBLE PRECISION UA,VA,UROOT,VROOT,EPSPX,UWORK,VWORK,UB,VB
245      DOUBLE PRECISION UPX,VPX,UDPX,UDPX,VPX,VPX,UBB,EPS
246      DOUBLE PRECISION DUM1,DUM2
247      DOUBLE PRECISION DSQRT
248      DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),UWORK(26),VWORK(26),MU
      ILTI(25),UB(26),VB(26)
249      COMMON DUM1,DUM2,EPS,IO2,MAX
250      DO 50 I=1,NR
251      MULTI(I)=0
252      KKK=NA+1
253      DO 20 J=1,KKK
254      UWORK(J)=UA(J)
255      20 VWORK(J)=VA(J)
256      NWORK=NA
257      25 CALL HORNER(NWORK,UWORK,VWORK,UROOT(I),VROOT(I),UB,VB,UPX,VPX)
258      NWORK=NWORK-1
259      BBB=DSQRT(UPX*UPX+VPX*VPX)
260      IF(BBB.GT.EPSPX) GO TO 45
261      MULTI(I)=MULTI(I)+1
262      IF(NWORK.LT.1) GO TO 50
263      KKK=NWORK+1
264      DO 40 J=1,KKK
265      UWORK(J)=UB(J)
266      40 VWORK(J)=VB(J)
267      GO TO 25
268      45 IF(MULTI(I).EQ.0) WRITE(IO2,1000) EPSPX,I,UROOT(I),VROOT(I)
269      50 CONTINUE
270      100 RETURN
271      1000 FORMAT(///14H THE EPSILON (,D10.03,49H) CHECK IN SUBROUTINE MULTIP
      1 INDICATES THAT ROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I/80H IS NO
      2T CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIP
      3LICITY 0//)
272      END

```

TABLE XXI-B (Continued)

```

273     SUBROUTINE MULLER(UA,VA,NP,UOIAPP,VOIAPP,NAPP,XSTART,XEND,EPSM,UP,
        LVP,NDEG,NOPOLY)
C     *****
C     *
C     * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
C     * POLYNOMIAL OF MAXIMUM DEGREE 25. THROUGH THREE GIVEN POINTS THE
C     * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC. THE ZERO OF THE QUADRATIC
C     * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.
C     * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.
C     *
C     *****
274     DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UROOT,VROOT,UX1,VX1,UAPP,VAPP
        1,UX2,VX2,UWORK,VWORK,UX3,VX3,UB,VB,UX4,VX4,UA,VA,UPX1,VPX1,URAPP,V
        2RAPP,UPX4,VPX4,EPSRT,EPSQ,EPS,CCC,EPMS,UH3,VH3,UQ4,VQ4,ABPX4,ABPX3,
        3,QQQ,XSTART,XEND
275     DOUBLE PRECISION DSQRT
276     DOUBLE PRECISION UOIAPP,VOIAPP
277     DOUBLE PRECISION UP,VP
278     DIMENSION UROOT(25),VROOT(25),MULT(25),UAPP(25,3),VAPP(25,3),UWORK
        1(26),VWORK(26),UB(26),VB(26),UA(26),VA(26),URAPP(25,3),VRAPP(25,3)
279     DIMENSION UOIAPP(25),VOIAPP(25)
280     DIMENSION UP(26),VP(26)
281     DATA PNAME,DNAME/2HP(,2HD(/
282     LOGICAL CONV
283     COMMON EPSRT,EPSQ,EPS,IO2,MAX
284     EPSRT=0.999
285     NROOT=0
286     IROOT=0
287     IPATH=1
288     NOMULT=0
289     NALTER=0
290     ITIME=0
291     IAPP=1
292     ITER=1
293     IF(NAPP.NE.0) GO TO 18
294     NAPP=NP
295     CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
296     GO TO 27
297     18 DO 20 I=1,NAPP
298         UAPP(I,2)=UOIAPP(I)
299     20 VAPP(I,2)=VOIAPP(I)
300         DO 25 I=1,NAPP
301             UAPP(I,1)=0.9*UAPP(I,2)
302             VAPP(I,1)=0.9*VAPP(I,2)
303             UAPP(I,3)=1.1*UAPP(I,2)
304             VAPP(I,3)=1.1*VAPP(I,2)
305     25 VAPP(I,3)=1.1*VAPP(I,2)
306     27 KKK=NP+1
307         DO 30 I=1,KKK
308             UWORK(I)=UA(I)
309             VWORK(I)=VA(I)
310             NWORK=NP
311         40 UX1=UAPP(IAPP,1)
312             VX1=VAPP(IAPP,1)
313             UX2=UAPP(IAPP,2)
314             VX2=VAPP(IAPP,2)
315             UX3=UAPP(IAPP,3)
316             VX3=VAPP(IAPP,3)
316     CALL HORNER(NWORK,UWORK,VWORK,UX1,VX1,UB,VB,UPX1,VPX1)
317     CALL HORNER(NWORK,UWORK,VWORK,UX2,VX2,UB,VB,UPX2,VPX2)
318     CALL HORNER(NWORK,UWORK,VWORK,UX3,VX3,UB,VB,UPX3,VPX3)

```

TABLE XXI-B (Continued)

```

319 50 CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
    14,VX4,UQ4,VQ4,UH3,VH3)
320 60 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
321 ABPX4=DSQRT(UPX4*UPX4+VPX4*VPX4)
322 ABPX3=DSQRT(UPX3*UPX3+VPX3*VPX3)
323 IF(ABPX3.EQ.0.0) GO TO 70
324 QQQ=ABPX4/ABPX3
325 IF(QQQ.LE.10.) GO TO 70
326 UQ4=0.5*UQ4
327 VQ4=0.5*VQ4
328 UX4=UX3+(UH3*UQ4-VH3*VQ4)
329 VX4=VX3+(VH3*UQ4+UH3*VQ4)
330 GO TO 60
331 70 CALL TEST(UX3,VX3,UX4,VX4,CONV)
332 IF(CONV) GO TO 120
333 IF(ITER.LT.MAX) GO TO 110
334 CALL ALTER(UAPP(IAPP,1),VAPP(IAPP,1),UAPP(IAPP,2),VAPP(IAPP,2),UAP
    1P(IAPP,3),VAPP(IAPP,3),NALTER,ITIME)
335 IF(NALTER.GT.5) GO TO 75
336 ITER=1
337 GO TO 40
338 75 IF(IAPP.LT.NAPP) GO TO 100
339 IF(XEND.EQ.0.0) GO TO 77
340 IF(XSTART.GT.XEND) GO TO 77
341 NAPP=NP
342 CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
343 IAPP=0
344 GO TO 100
345 77 WRITE(IO2,1090)
346 KKK=NWORK+1
347 WRITE(IO2,1035) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
348 80 IF(NROOT.EQ.0) GO TO 90
349 WRITE(IO2,1060)
350 IF(IPATH.EQ.1) GO TO 82
351 81 IPATH=2
352 CALL BETTER(UA,VA,NP,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MULT)
353 WRITE(IO2,1200)
354 CALL MULTIP(UP,VP,NDEG,UROOT,VROOT,NROOT,MULT,EPSM)
355 82 IF(NROOT.EQ.0) GO TO 90
356 IF(IROOT.EQ.0) GO TO 85
357 WRITE(IO2,1080)
358 DO 55 I=1,IROOT
359 55 WRITE(IO2,1085) I,UROOT(I),VROOT(I),MULT(I),URAPP(I,2),VRAPP(I,2)
360 IF(IROOT.LT.NROOT) GO TO 85
361 GO TO 87
362 85 KKK=IROOT+1
363 WRITE(IO2,1086) (I,UROOT(I),VROOT(I),MULT(I),I=KKK,NROOT)
364 87 IF(IPATH.EQ.1) GO TO 81
365 RETURN
366 90 WRITE(IO2,1070) NOPOLY
367 RETURN
368 100 IAPP=IAPP+1
369 ITER=1
370 NALTER=0
371 GO TO 40
372 120 NROOT=NROOT+1
373 IROOT=NROOT
374 MULT(NROOT)=1
375 NOMULT=NOMULT+1
376 UROOT(NROOT)=UX4

```

TABLE XXI-B (Continued)

```

377      VROOT(NROOT)=VX4
378      URAPP(NROOT,1)=UAPP(IAPP,1)
379      VRAPP(NROOT,1)=VAPP(IAPP,1)
380      URAPP(NROOT,2)=UAPP(IAPP,2)
381      VRAPP(NROOT,2)=VAPP(IAPP,2)
382      URAPP(NROOT,3)=UAPP(IAPP,3)
383      VRAPP(NROOT,3)=VAPP(IAPP,3)
384      125 IF(NOMULT.LT.NP) GO TO 130
385          GO TO 80
386      130 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
387          NWORK=NWORK-1
388          KKK=NWORK+1
389          DO 140 I=1,KKK
390              UWORK(I)=UB(I)
391      140 VWORK(I)=VB(I)
392          CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
393          CCC=DSQRT(UPX4*UPX4+VPX4*VPX4)
394          IF(CCC.LT.EPSM) GO TO 150
395          IF(NWORK.GT.2) GO TO 75
396          IROOT=NROOT
397          CALL QUAD(UWORK,VWORK,NWORK,UROOT,VROOT,NROOT,MULT,EPSO)
398          GO TO 80
399      150 MULT(NROOT)=MULT(NROOT)+1
400          NOMULT=NOMULT+1
401          GO TO 125
402      110 UX1=UX2
403          VX1=VX2
404          UX2=UX3
405          VX2=VX3
406          UX3=UX4
407          VX3=VX4
408          UPX1=UPX2
409          VPX1=VPX2
410          UPX2=UPX3
411          VPX2=VPX3
412          UPX3=UPX4
413          VPX3=VPX4
414          ITER=ITER+1
415          GO TO 50
416      1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
          1ZEROS WERE FOUND//)
417      1080 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
          1 APPROXIMATION//)
418      1070 FORMAT(//,43H NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
419      1086 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,8X,I2,9X,23HS
          1OLVED BY DIRECT METHOD)
420      1035 FORMAT(3X,A2,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
421      1085 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,8X,I2,8X,D23.
          116,3H + ,D23.16,2H I)
422      1000 FORMAT(3(I2,1X),9X,I3,8X,3(D6.0,1X),13X,2(D7.0,1X),I1)
423      1060 FORMAT(///35H BEFORE ATTEMPT TO IMPROVE ACCURACY)
424      1200 FORMAT(///1X,37HAFTER THE ATTEMPT TO IMPROVE ACCURACY)
425      END

```

TABLE XXI-B (Continued)

```

426      SUBROUTINE ALTER(X1R,X1I,X2R,X2I,X3R,X3I,NALTER,ITIME)
C      *****
C      *
C      * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO
C      * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.
C      *
C      *****
427      DOUBLE PRECISION X1R,X1I,X2R,X2I,X3R,X3I,EPS1,EPS2,EPS3,R,BETA
428      DOUBLE PRECISION DCOS,DSIN
429      DOUBLE PRECISION DATAN2
430      DOUBLE PRECISION DSQRT
431      COMMON EPS1,EPS2,EPS3,I02,MAX
432      IF(ITIME.NE.0) GO TO 5
433      ITIME=1
434      WRITE(I02,1010) MAX
435      5 IF(NALTER.EQ.0) GO TO 10
436      WRITE(I02,1000) X1R,X1I,X2R,X2I,X3R,X3I
437      GO TO 20
438      10 R=DSQRT(X2R*X2R+X2I*X2I)
439      BETA=DATAN2(X2I,X2R)
440      WRITE(I02,1020) X1R,X1I,X2R,X2I,X3R,X3I
441      20 NALTER=NALTER+1
442      IF(NALTER.GT.5) RETURN
443      GO TO (30,40,30,40,30),NALTER
444      30 X2R=-X2R
445      X2I=-X2I
446      GO TO 50
447      40 BETA=BETA+1.0471976
448      X2R=R*DCOS(BETA)
449      X2I=R*DSIN(BETA)
450      50 X1R=0.9*X2R
451      X1I=0.9*X2I
452      X3R=1.1*X2R
453      X3I=1.1*X2I
454      RETURN
455      1000 FORMAT(1X,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22HALTERED APPROXIM
LATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
2,D23.16,2H I/)
456      1020 FORMAT(1H0,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22INITIAL APPROXI
MATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
2 ,D23.16,2H I/)
457      1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
TER ,I3,12H ITERATIONS.//)
458      END

```

TABLE XXI-B (Continued)

```

459      SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
C      *****
C      *
C      * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE. *
C      * DEGREE OF THE ORIGINAL POLYNOMIAL. *
C      *
C      *****
460      DOUBLE PRECISION APPR,APPI,XSTART,EPS1,EPS2,EPS3,BETA
461      DOUBLE PRECISION DCOS,DSIN
462      DIMENSION APPR(25,3),APPI(25,3)
463      COMMON EPS1,EPS2,EPS3,IO2,MAX
464      IF(XSTART.EQ.0.0) XSTART=0.5
465      BETA=0.2617994
466      DO 10 I=1,NAPP
467      APPR(I,2)=XSTART*DCOS(BETA)
468      APPI(I,2)=XSTART*DSIN(BETA)
469      BETA=BETA+0.5235988
470      10 XSTART=XSTART+0.5
471      DO 20 I=1,NAPP
472      APPR(I,1)=0.9*APPR(I,2)
473      APPI(I,1)=0.9*APPI(I,2)
474      APPR(I,3)=1.1*APPR(I,2)
475      20 APPI(I,3)=1.1*APPI(I,2)
476      RETURN
477      END

```

TABLE XXI-B (Continued)

```

478     SUBROUTINE BETTER(UA,VA,NP,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MUL
C      1T)
C      *****
C      *
C      * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND *
C      * BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO *
C      * THE FULL, UNDEFLATED POLYNOMIAL. *
C      *
C      *****
479     DOUBLE PRECISION UROOT,VROOT,UA,VA,UBAPP,VBAPP,UX1,VX1,UX2,VX2,UX3
1,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UB,VB,UROOTS,VROOTS,EPSRT,UX4,V
2X4,URAPP,VRAPP,EPSO,EPS,UQ4,VQ4,UH3,VH3
480     LOGICAL CONV
481     DIMENSION UROOT(25),VROOT(25),UA(26),VA(26),UBAPP(25,3),VBAPP(25,3
1),UB(26),VB(26),UROOTS(25),VROOTS(25),URAPP(25,3),VRAPP(25,3),MULT
3(25)
482     COMMON EPSRT,EPSO,EPS,IO2,MAX
483     IF(NROOT.LE.1) RETURN
484     L=0
485     DO 10 I=1,NROOT
486     UBAPP(I,1)=UROOT(I)*EPSRT
487     VBAPP(I,1)=VROOT(I)*EPSRT
488     UBAPP(I,2)=UROOT(I)
489     VBAPP(I,2)=VROOT(I)
490     UBAPP(I,3)=UROOT(I)*(2.0-EPSRT)
491 10  VBAPP(I,3)=VROOT(I)*(2.0-EPSRT)
492     DO 100 J=1,NROOT
493     UX1=UBAPP(J,1)
494     VX1=VBAPP(J,1)
495     UX2=UBAPP(J,2)
496     VX2=VBAPP(J,2)
497     UX3=UBAPP(J,3)
498     VX3=VBAPP(J,3)
499     ITER=1
500     CALL HORNER(NP,UA,VA,UX1,VX1,UB,VB,UPX1,VPX1)
501     CALL HORNER(NP,UA,VA,UX2,VX2,UB,VB,UPX2,VPX2)
502 20  CALL HORNER(NP,UA,VA,UX3,VX3,UB,VB,UPX3,VPX3)
503     CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
14,VX4,UQ4,VQ4,UH3,VH3)
504 30  CALL TEST(UX3,VX3,UX4,VX4,CONV)
505     IF(CONV) GO TO 50
506     IF(ITER.LT.MAX) GO TO 40
507     WRITE(IO2,1000) J,UROOT(J),VROOT(J),MAX
508     WRITE(IO2,1010) UX4,VX4
509     IF(J.LT.IROOT) GO TO 33
510     IF(J.EQ.IROOT) GO TO 35
511     GO TO 100
512 33  KKK=IROOT-1
513     DO 34 K=J,KKK
514     URAPP(K,1)=URAPP(K+1,1)
515     VRAPP(K,1)=VRAPP(K+1,1)
516     URAPP(K,2)=URAPP(K+1,2)
517     VRAPP(K,2)=VRAPP(K+1,2)
518     URAPP(K,3)=URAPP(K+1,3)
519 34  VRAPP(K,3)=VRAPP(K+1,3)
520 35  IROOT=IROOT-1
521     GO TO 100
522 40  UX1=UX2
523     VX1=VX2
524     UX2=UX3

```


TABLE XXI-B (Continued)

```

525     VX2=VX3
526     UX3=UX4
527     VX3=VX4
528     UPX1=UPX2
529     VPX1=VPX2
530     UPX2=UPX3
531     VPX2=VPX3
532     ITER=ITER+1
533     GO TO 20
534 50  L=L+1
535     UROOTS(L)=UX4
536     VROOTS(L)=VX4
537     MULT(L)=MULT(J)
538 100  CONTINUE
539     IF(L.EQ.0) GO TO 120
540     DO 110 I=1,L
541     UROOT(I)=UROOTS(I)
542 110  VROOT(I)=VROOTS(I)
543     NROOT=L
544     RETURN
545 120  NROOT=0
546     RETURN
547 1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,
1D23.16,3H + ,D23.16,2H I/24H DID NOT CONVERGE AFTER ,I3,11H ITERAT
2IONS)
548 1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,D23.16,3H + ,D23.16,2H I/
1/)
549     END

```

TABLE XXI-B (Continued)

```

550      SUBROUTINE CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,V
      1PX3,UX4,VX4,UQ4,VQ4,UH3,VH3)
C      *****
C      *
C      * GIVEN THREE APPROXIMATIONS X(N-2), X(N-1), AND X(N), SUBROUTINE CALC
C      * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF
C      * THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION
C      * X(N+1) TO THE ZERO OF THE POLYNOMIAL.
C      *
C      *****
551      DOUBLE PRECISION ARG1,ARG2
552      DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UX1,VX1,UX2,VX2,UX3,VX3,UPX1,
      1VPX1,UH3,VH3,UH2,VH2,UQ3,VQ3,UD,VD,UB,VB,UC,VC,UDISC,VDISC,UCCC,VC
      2CC,UDEN1,VDEN1,UDEN2,VDEN2,UQ4,VQ4,UX4,VX4,EPSRT,EPSO,EPS,UDDD,VDD
      3D,AAA,BBB,RAD,UAAA,VAAA,UBBB,VBBB
553      DOUBLE PRECISION THETA,ANGLE,UTEST,VTEST
554      DOUBLE PRECISION DATAN2,DCOS,DSIN,DSQRT
555      COMMON EPSRT,EPSO,EPS,IO2,MAX
556      UH3=UX3-UX2
557      VH3=VX3-VX2
558      UH2=UX2-UX1
559      VH2=VX2-VX1
560      BBB=UH2*UH2+VH2*VH2
561      UQ3=(UH3*UH2+VH3*VH2)/BBB
562      VQ3=(VH3*UH2-UH3*VH2)/BBB
563      UDDD=1.0+UQ3
564      VDDD=VQ3
565      UD=(UPX3-(UDDD*UPX2-VDDD*VPX2))+(UQ3*UPX1-VQ3*VPX1)
566      VD=(VPX3-(VDDD*UPX2+UDDD*VPX2))+(VQ3*UPX1+UQ3*VPX1)
567      UAAA=2.0*UQ3
568      VAAA=2.0*VQ3
569      UAAA=UAAA+1.0
570      UBBB=UDDD*UDDD-VDDD*VDDD
571      VBBB=VDDD*UDDD+UDDD*VDDD
572      UCCC=UQ3*UQ3-VQ3*VQ3
573      VCCC=VQ3*UQ3+UQ3*VQ3
574      UB=(UAAA*UPX3-VAAA*VPX3)-(UBBB*UPX2-VBBB*VPX2)+(UCCC*UPX1-VCCC*V
      1PX1)
575      VB=(VAAA*UPX3+UAAA*VPX3)-(VBBB*UPX2+UBBB*VPX2)+(VCCC*UPX1+UCCC*V
      1PX1)
576      UC=UDDD*UPX3-VDDD*VPX3
577      VC=VDDD*UPX3+UDDD*VPX3
578      UDISC=(UB*UB-VB*VB)-(4.0*(UD*UC-VD*VC))
579      VDISC=(2.0*(VB*UB))-(4.0*(VD*UC+UD*VC))
580      AAA=DSQRT(UDISC*UDISC+VDISC*VDISC)
581      IF(AAA.EQ.0.0) GO TO 5
582      GO TO 7
583      5 THETA=0.0
584      GO TO 9
585      7 THETA=DATAN2(VDISC,UDISC)
586      9 RAD=DSQRT(AAA)
587      ANGLE=THETA/2.0
588      UTEST=RAD*DCOS(ANGLE)
589      VTEST=RAD*DSIN(ANGLE)
590      UDEN1=UB+UTEST
591      VDEN1=VB+VTEST
592      UDEN2=UB-UTEST
593      VDEN2=VB-VTEST
594      ARG1=UDEN1*UDEN1+VDEN1*VDEN1
595      ARG2=UDEN2*UDEN2+VDEN2*VDEN2

```

TABLE XXI-B (Continued)

```

596     AAA=DSQRT(ARG1)
597     BBB=DSQRT(ARG2)
598     IF(AAA.LT.BBB) GO TO 10
599     IF(AAA.EQ.0.0) GO TO 60
600     UAAA=-2.0*UC
601     VAAA=-2.0*VC
602     UQ4=(UAAA*UDEN1+VAAA*VDEN1)/ARG1
603     VQ4=(VAAA*UDEN1-UAAA*VDEN1)/ARG1
604     GO TO 50
605 10  IF(BBB.EQ.0.0) GO TO 60
606     UAAA=-2.0*UC
607     VAAA=-2.0*VC
608     UQ4=(UAAA*UDEN2+VAAA*VDEN2)/ARG2
609     VQ4=(VAAA*UDEN2-UAAA*VDEN2)/ARG2
610     GO TO 50
611 50  UX4=UX3+(UH3*UQ4-VH3*VQ4)
612     VX4=VX3+(VH3*UQ4+UH3*VQ4)
613     RETURN
614 60  UQ4=1.0
615     VQ4=0.0
616     GO TO 50
617     END

```

```

618     SUBROUTINE TEST(UX3,VX3,UX4,VX4,CONV)
C *****
C *
C * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
C * IMATIONS BY TESTING THE EXPRESSION
C * ABSOLUTE VALUE OF (X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).
C * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
C *
C *****
619     DOUBLE PRECISION UX3,VX3,UX4,VX4,EPSRT,EPSO,EPS,AAA,UDUMMY,VDUMMY,
1DENOM
620     DOUBLE PRECISION DSQRT
621     LOGICAL CONV
622     COMMON EPSRT,EPSO,EPS,IO2,MAX
623     UDUMMY=UX4-UX3
624     VDUMMY=VX4-VX3
625     AAA=DSQRT(UDUMMY*UDUMMY+VDUMMY*VDUMMY)
626     DENOM=DSQRT(UX4*UX4+VX4*VX4)
627     IF(DENOM.LT.EPSO) GO TO 20
628     IF(AAA/DENOM.LT.EPS) GO TO 10
629     5 CONV=.FALSE.
630     GO TO 100
631     10 CONV=.TRUE.
632     GO TO 100
633     20 IF(AAA.LT.EPSO) GO TO 10
634     GO TO 5
635     100 RETURN
636     END

```

TABLE XXI-B (Continued)

```

637      SUBROUTINE HORNER(NA,UA,VA,UX,VX,UB,VB,UPX,VPX)
C      *****
C      *
C      * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D. *
C      * SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE *
C      * FACTOR (X-D). *
C      *
C      *****
638      DOUBLE PRECISION UX,VX,UPX,VPX,UB,VB,UA,VA
639      DIMENSION UA(26),VA(26),UB(26),VB(26)
640      UB(1)=UA(1)
641      VB(1)=VA(1)
642      NUM=NA+1
643      DO 10 I=2,NUM
644      UB(I)=UA(I)+(UB(I-1)*UX-VB(I-1)*VX)
645 10  VB(I)=VA(I)+(VB(I-1)*UX+UB(I-1)*VX)
646      UPX=UB(NUM)
647      VPX=VB(NUM)
648      RETURN
649      END

650      SUBROUTINE COMSQT(UX,VX,UY,VY)
C      *****
C      *
C      * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER. *
C      *
C      *****
651      DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
652      DOUBLE PRECISION DSQRT,DABS
653      R=DSQRT(UX*UX+VX*VX)
654      AAA=DSQRT(DABS((R+UX)/2.0))
655      BBB=DSQRT(DABS((R-UX)/2.0))
656      IF(VX) 10,20,30
657 10  UY=AAA
658      VY=-1.0*BBB
659      GO TO 100
660 20  IF(UX) 40,50,60
661 30  UY=AAA
662      VY=BBB
663      GO TO 100
664 40  DUMMY=DABS(UX)
665      UY=0.0
666      VY=DSQRT(DUMMY)
667      GO TO 100
668 50  UY=0.0
669      VY=0.0
670      GO TO 100
671 60  DUMMY=DABS(UX)
672      UY=DSQRT(DUMMY)
673      VY=0.0
674 100 RETURN
675      END

```

\$ENTRY

APPENDIX F

POLYNOMIAL GENERATOR

1. Use of the Programs

Given N linear factors of the form $(AX + B)$ where A and B are complex numbers, the polynomial generator program computes their product resulting in a polynomial of degree N of the form

$$P(X) = a_1 X^N + a_2 X^{N-1} + \dots + a_N X + a_{N+1}.$$

Note that if $(AX + B)$ is a factor of the polynomial $P(X)$, then $\frac{-B}{A}$ is a zero of $P(X)$.

Two programs of the polynomial generator are presented here. The first is the single precision program. The second program is in double precision and is designed to perform double precision complex arithmetic. These programs are written for use on any computer using FORTRAN IV language. They have been tested on the IBM S/360 mod. 50 computer which has a 32 bit word.

Each program is designed to construct polynomials of degree 25 or less. In order to construct polynomials of degree N where $N > 25$, the array dimensions must be changed as listed in Table XXII for both single precision and double precision programs.

TABLE XXII

ARRAYS USED IN THE POLYNOMIAL GENERATOR

<u>Single Precision</u>		<u>Double Precision</u>
	Main Program	
FACTOR(N,2) C(N+1)		FACTOR(N,2), FACTOI(N,2) CR(N+1), CI(N+1)
	Subroutine POLY	
FACTOR(N,2) C(N+1) D(2)		FACTOR(N,2), FACTOI(N,2) CR(N+1), CI(N+1) DR(2), DI(2)

The single precision program may be converted to double precision for use on machines equipped to perform double precision complex arithmetic provided the following changes or their equivalent are made: The changes presented below are those required for the IBM S/360.

1. The format statements should be changed from E-type to D-type.
2. Change COMPLEX C_1, C_2, \dots to COMPLEX*16 C_1, C_2, \dots where C_1, C_2, \dots are complex variables.
3. Add IMPLICIT REAL*8(A-H,O-Z)

After selecting the desired program, the input data should be prepared as described in section 2.

2. Input Data for the Polynomial Generator

The input data for the polynomial generator is grouped into

polynomial data sets. Each polynomial data set consists of the data to generate one and only one polynomial. As many polynomials as the user desires may be generated by placing the polynomial data sets one behind the other. Each polynomial data set consists of two kinds of information placed in the following order:

1. Control information
2. Factors of the polynomial

An end card follows the entire collection of data sets. It indicates that there is no more data to follow and terminates execution of the program. This information is displayed in figure 17 and described below. For single precision data, the E-type specification should be used, while for double precision data, the D-type specification should be used. All data should be right justified.

Control Information

The control card is the first card of the polynomial data set and contains the information given in Table XXIII. See figure 14.

TABLE XXIII

CONTROL DATA FOR THE POLYNOMIAL GENERATOR

<u>Variable Name</u>	<u>Card Columns</u>	<u>Description</u>
N	c.c. 1-2	Number of linear factors to be multiplied together. Integer. Right justify.
KCHECK	c.c. 80	This should be left blank.

Factors of the Polynomial

The factors to be read in should be of the form $(AX + B)$ where A and B are non-zero complex numbers. The coefficient of the X term, A, is entered first. Both coefficients of a factor are entered on the same card as described in Table XXIV and illustrated in figure 15. The variable in parentheses is the double precision equivalent. The example in figure 15 is $X + (2.5 - 7i)$.

TABLE XXIV

FACTOR DATA FOR THE POLYNOMIAL GENERATOR

<u>Variable Name</u>	<u>Card Columns</u>	<u>Description</u>
FACTOR(I,1) (FACTOR(I,1))	c.c. 1-20	Real part of the complex number A. Real. Right justify.
FACTOR(I,1) (FACTOI(I,1))	c.c. 21-40	Imaginary part of the complex number A. Real. Right justify.
FACTOR(I,2) (FACTOR(I,2))	c.c. 41-60	Real part of the complex number B. Real. Right justify.
FACTOR(I,2) (FACTOI(I,2))	c.c. 61-80	Imaginary part of the complex number B. Real. Right justify.

End Card

The end card is the last card of the input data to the program.

It indicates that there is no more data to be read. When this card is read, program execution is terminated. This card is described in Table XXV and illustrated in figure 16.

TABLE XXV

DATA TO END EXECUTION OF THE POLYNOMIAL GENERATOR

<u>Variable Name</u>	<u>Card Columns</u>	<u>Description</u>
KCHECK	c.c. 80	Must contain the number 1. Integer.

3. Variables Used in the Polynomial Generator

The definitions of the major variables used in the polynomial generator are given in Table XXVI. The notation and symbols used are described in Appendix B, § 3. Variables not listed are dummy variables used for temporary storage of computations.

4. Description of Program Output

The output from the polynomial generator consists of the following information and is illustrated in Exhibit DD.

The linear factors are printed, in the order read, under the heading "THE XX FACTORS OF THE POLYNOMIAL TO BE GENERATED ARE." XX is the number of linear factors given. The form of each factor is $AX + B$.

The coefficients of the resulting polynomial are printed,

coefficient of highest degree term first, under the heading "THE POLYNOMIAL OF DEGREE XX GENERATED FROM THE ABOVE FACTORS IS." XX is the degree of the polynomial.

000000000111111111122222222233333333334444444445555555556666666667777777778				
12345678901234567890123456789012345678901234567890123456789012345678901234567890				
<p style="text-align: center;">FACTOR(I,1) (FACTOR(I,1))</p>	<p style="text-align: center;">FACTOR(I,1) (FACTOI(I,1))</p>	<p style="text-align: center;">FACTOR(I,2) (FACTOR(I,2))</p>	<p style="text-align: center;">FACTOR(I,2) (FACTOI(I,2))</p>	
<p style="text-align: right;">0.1D+01</p> <p style="text-align: right;">0.1E+01</p>	<p style="text-align: right;">0.0D+00</p> <p style="text-align: right;">0.0E+00</p>	<p style="text-align: right;">0.25D+01</p> <p style="text-align: right;">0.25E+01</p>	<p style="text-align: right;">-0.7D+01</p> <p style="text-align: right;">-0.7E+01</p>	

Figure 15. Factor Card for the Polynomial Generator

0000000001111111112222222223333333334444444445555555556666666667777777778	K
1234567890123456789012345678901234567890123456789012345678901234567890	C
	H
	E
	C
	K
	L
	L

Figure 16. End Card for the Polynomial Generator

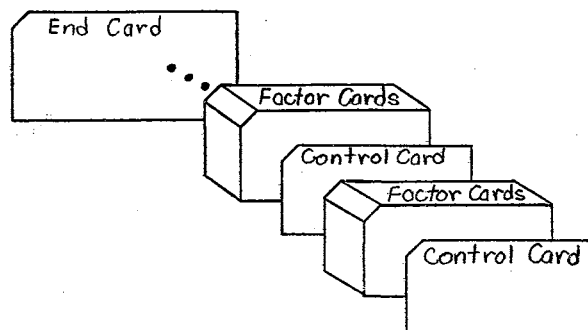


Figure 17. Sequence of Input Data for the Polynomial Generator

TABLE XXVI

VARIABLES USED IN THE POLYNOMIAL GENERATOR

<u>Single Precision</u> <u>Variable</u>	<u>Type</u>	<u>Double Precision</u> <u>Variable</u>	<u>Type</u>	<u>Disposition</u> <u>of Argument</u>	<u>Description</u>
Main Program					
I01	I	I01	I		Unit number of the input device
I02	I	I02	I		Unit number of the output device
N	I	N	I		Number of factors to be read
KCHECK	I	KCHECK	I		KCHECK = 1 implies that execution will be terminated
FACTOR	C	FACTOR, FACTOI	R		Array containing the coefficients of the linear factors
C	C	CR, CI	R		Array containing coefficients of the polynomial
Subroutine POLY					
NC	I	NC	I		Counter
C	C	CR, CI	R	R	Array of coefficients of polynomial
FACTOR	C	FACTOR, FACTOI	R	E	Array of factors
N	I	N	I	E	Number of factors to be multiplied

THE 7 FACTORS OF THE POLYNOMIAL TO BE GENERATED ARE

```
FACTOR1 1) = 0.1000000000000000 01 + 0.0000000000000000 00 I X + -0.1000000000000000 01 + -0.1000000000000000 01 I
FACTOR1 2) = 0.1000000000000000 01 + 0.0000000000000000 00 I X + -0.2000000000000000 01 + 0.3000000000000000 01 I
FACTOR1 3) = 0.1000000000000000 01 + 0.0000000000000000 00 I X + 0.1000000000000000 01 + 0.0000000000000000 00 I
FACTOR1 4) = 0.1000000000000000 01 + 0.0000000000000000 00 I X + 0.0000000000000000 00 + -0.2000000000000000 01 I
FACTOR1 5) = 0.1000000000000000 01 + 0.0000000000000000 00 I X + 0.3000000000000000 01 + -0.3000000000000000 01 I
FACTOR1 6) = 0.1000000000000000 01 + 0.0000000000000000 00 I X + 0.1000000000000000 01 + 0.1000000000000000 01 I
FACTOR1 7) = 0.1000000000000000 01 + 0.0000000000000000 00 I X + 0.1000000000000000 01 + 0.1000000000000000 01 I
```

THE POLYNOMIAL OF DEGREE 7 GENERATED FROM THE ABOVE FACTORS IS

```
CI 1) = 0.1000000000000000 01 + 0.0000000000000000 00 I
CI 2) = 0.3000000000000000 01 + -0.1000000000000000 01 I
CI 3) = 0.8000000000000000 01 + 0.9000000000000000 01 I
CI 4) = 0.2399999999999999 02 + 0.1600000000000000 02 I
CI 5) = 0.7799999999999999 02 + 0.1800000000000000 02 I
CI 6) = 0.7999999999999999 02 + -0.2800000000000000 02 I
CI 7) = 0.6799999999999999 02 + -0.1120000000000000 03 I
CI 8) = 0.4800000000000000 02 + -0.7200000000000000 02 I
```

COMPILE TIME= 1.96 SEC, EXECUTION TIME= 0.49 SEC, OBJECT CODE= 3472 BYTES, ARRAY AREA= 1248 BYTES, UNUSED= 65280 BYTES

COMPILE TIME= 0.11 SEC, EXECUTION TIME= 0.00 SEC, OBJECT CODE= 3472 BYTES, ARRAY AREA= 1248 BYTES, UNUSED= 65280 BYTES

\$\$STOP

Exhibit DD. Linear Factors Are: $X + (-1 - i)$, $X + (-2 + 3i)$, $X + 1$, $X + (-2i)$,
 $X + (3 - 3i)$, $X + (1 + i)$, $X + (1 + i)$.

MAIN PROGRAM

POLY

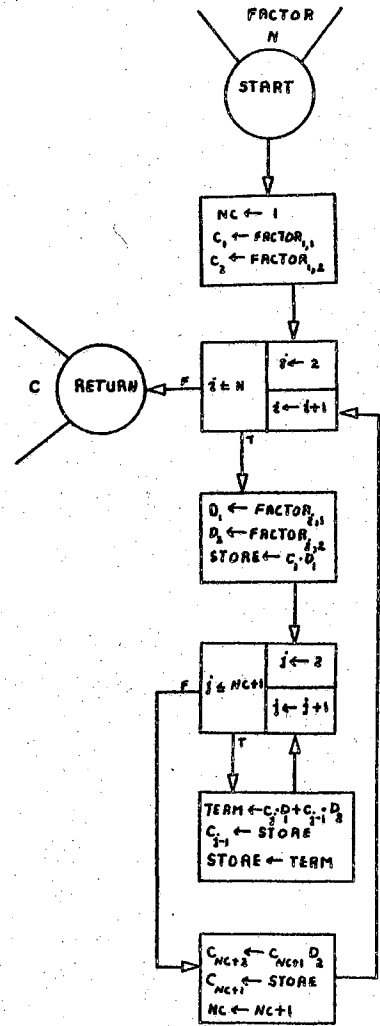
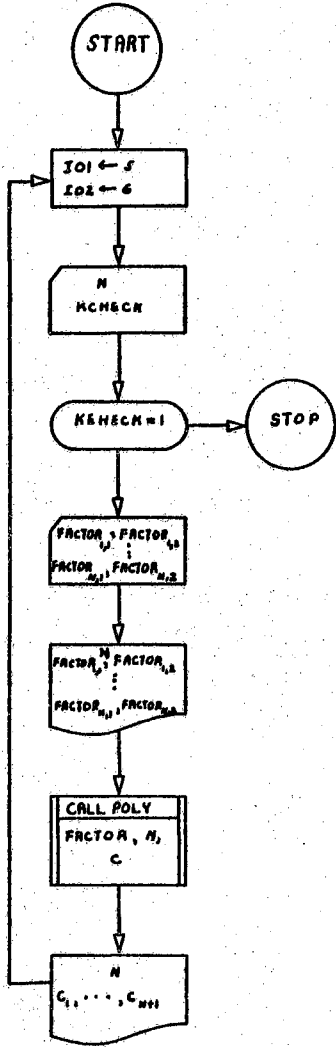


Figure 18. Flow Charts for the Polynomial Generator

TABLE XXVII

SINGLE PRECISION PROGRAM FOR THE POLYNOMIAL GENERATOR

```

$JOB 10414

C *****
C *
C * SINGLE PRECISION PROGRAM FOR THE POLYNOMIAL GENERATOR.
C *
C *
C * THIS PROGRAM CONSTRUCTS THE N-TH DEGREE POLYNOMIAL FORMED BY
C * MULTIPLYING N GIVEN LINEAR FACTORS TOGETHER.
C *
C *****

1  COMPLEX FACTOR,C
2  DIMENSION FACTOR(25,2),C(26)
3  IO1=5
4  IO2=6
5  1 READ(IO1,1000) N,KCHECK
6  IF(KCHECK.EQ.1) STOP
7  READ(IO1,1010) (FACTOR(I,1),FACTOR(I,2),I=1,N)
8  WRITE(IO2,1020) N
9  WRITE(IO2,1030) (I,FACTOR(I,1),FACTOR(I,2),I=1,N)
10 CALL POLY(FACTOR,N,C)
11 WRITE(IO2,1040) N
12 KKK=N+1
13 WRITE(IO2,1050) (I,C(I),I=1,KKK)
14 GO TO 1
15 1000 FORMAT(I2,77X,I1)
16 1010 FORMAT(4E20.0)
17 1020 FORMAT(1H1,4HTHE ,I2,46H FACTORS OF THE POLYNOMIAL TO BE GENERATED
18 1 ARE//)
18 1030 FORMAT(1X,7HFACTOR(,I2,4H) = ,E14.7,3H + ,E14.7,5H I X,7H +
19 1E14.7,3H + ,E14.7,2H I)
19 1040 FORMAT(///1X,25HTHE POLYNOMIAL OF DEGREE ,I2,36H GENERATED FROM TH
20 1E ABOVE FACTORS IS//)
20 1050 FORMAT(1X,2HC(,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
21 END

22 SUBROUTINE POLY(FACTOR,N,C)
C *****
C *
C * GIVEN N LINEAR FACTORS OF THE FORM (AX+B) WHERE A AND B ARE COMPLEX,
C * SUBROUTINE POLY FORMS THEIR PRODUCT.
C *
C *****

23 COMPLEX FACTOR,C,D,STORE,TERM
24 DIMENSION FACTOR(25,2),C(26),D(2)
25 NC=1
26 C(1)=FACTOR(1,1)
27 C(2)=FACTOR(1,2)
28 DO 50 I=2,N
29 D(1)=FACTOR(I,1)
30 D(2)=FACTOR(I,2)
31 STORE=C(1)*D(1)
32 KKK=NC+1
33 DO 20 J=2,KKK
34 TERM=(C(J)*D(1))+(C(J-1)*D(2))
35 C(J-1)=STORE
36 20 STORE=TERM
37 C(NC+2)=C(NC+1)*D(2)
38 C(NC+1)=STORE
39 50 NC=NC+1
40 RETURN
41 END

```

TABLE XXVIII

DOUBLE PRECISION PROGRAM FOR THE POLYNOMIAL GENERATOR

```

$JOB 10414
C *****
C *
C * DOUBLE PRECISION PROGRAM FOR THE POLYNOMIAL GENERATOR.
C *
C *
C * THIS PROGRAM CONSTRUCTS THE N-TH DEGREE POLYNOMIAL FORMED BY
C * MULTIPLYING N GIVEN LINEAR FACTORS TOGETHER.
C *
C *****
1  DOUBLE PRECISION FACTOR,FACTOI,CR,CI
2  DIMENSION FACTOR(25,2),FACTOI(25,2),CR(26),CI(26)
3  IO1=5
4  IO2=6
5  1 READ(IO1,1000) N,KCHECK
6  IF(KCHECK.EQ.1) STOP
7  READ(IO1,1010) (FACTOR(I,1),FACTOI(I,1),FACTOR(I,2),FACTOI(I,2),I=
8  11,N)
9  WRITE(IO2,1020) N
10 WRITE(IO2,1030) (I,FACTOR(I,1),FACTOI(I,1),FACTOR(I,2),FACTOI(I,2)
11 1,I=1,N)
10 CALL POLY(FACTOR,FACTOI,N,CR,CI)
11 WRITE(IO2,1040) N
12 KKK=N+1
13 WRITE(IO2,1050) (I,CR(I),CI(I),I=1,KKK)
14 GO TO 1
15 1000 FORMAT(I2,77X,I1)
16 1010 FORMAT(4D20.0)
17 1020 FORMAT(1H1,4HTHE ,I2,46H FACTORS OF THE POLYNOMIAL TO BE GENERATED
18 1 ARE//)
18 1030 FORMAT(1X,7HFACTOR(,I2,4H) = ,D23.16,3H + ,D23.16,5H I X,7H +
19 1 ,D23.16,3H + ,D23.16,2H I)
19 1040 FORMAT(///1X,25HTHE POLYNOMIAL OF DEGREE ,I2,36H GENERATED FROM TH
20 1E ABOVE FACTORS IS//)
20 1050 FORMAT(1X,2HC(,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
21 END

```

TABLE XXVIII (Continued)

```

22      SUBROUTINE POLY(FACTOR,FACTOI,N,CR,CI)
C      *****
C      *
C      * GIVEN N LINEAR FACTORS OF THE FORM (AX+B) WHERE A AND B ARE COMPLEX,
C      * SUBROUTINE POLY FORMS THEIR PRODUCT.
C      *
C      *****
23      DOUBLE PRECISION FACTOR,FACTOI,CR,CI,STORER,STOREI,TERMR,TERMI,DR,
        1DI
24      DIMENSION FACTOR(25,2),FACTOI(25,2),CR(26),CI(26),DR(2),DI(2)
25      NC=1
26      CR(1)=FACTOR(1,1)
27      CI(1)=FACTOI(1,1)
28      CR(2)=FACTOR(1,2)
29      CI(2)=FACTOI(1,2)
30      DO 50 I=2,N
31      DR(1)=FACTOR(I,1)
32      DI(1)=FACTOI(I,1)
33      DR(2)=FACTOR(I,2)
34      DI(2)=FACTOI(I,2)
35      STORER=CR(1)*DR(1)-CI(1)*DI(1)
36      STOREI=CI(1)*DR(1)+CR(1)*DI(1)
37      KKK=NC+1
38      DO 20 J=2,KKK
39      TERMR=(CR(J)*DR(1)-CI(J)*DI(1))+(CR(J-1)*DR(2)-CI(J-1)*DI(2))
40      TERMI=(CI(J)*DR(1)+CR(J)*DI(1))+(CI(J-1)*DR(2)+CR(J-1)*DI(2))
41      CR(J-1)=STORER
42      CI(J-1)=STOREI
43      STORER=TERMR
44      STOREI=TERMI
20      CR(NC+2)=CR(NC+1)*DR(2)-CI(NC+1)*DI(2)
45      CI(NC+2)=CI(NC+1)*DR(2)+CR(NC+1)*DI(2)
46      CR(NC+1)=STORER
47      CI(NC+1)=STOREI
48      50 NC=NC+1
49      RETURN
50      END
51

```

\$ENTRY

VITA

2

Randy Joe Snider

Candidate for the Degree of

Master of Science

Thesis: NUMERICALLY APPROXIMATING ZEROS OF A POLYNOMIAL

Major Field: Mathematics

Biographical:

Personal Data: Born in Oklahoma City, Oklahoma, August 16, 1945, the son of Bill M. and Dorothy Elizabeth Snider.

Education: Graduated from Antlers High School, Antlers, Oklahoma, in May, 1963; received the Bachelor of Science degree with a major in mathematics from Oklahoma State University in 1968; completed requirements for Master of Science degree at Oklahoma State University in May, 1970.

Professional Experience: Graduate Assistant, Mathematics Department, Oklahoma State University, 1968 and 1969; Programmer, Skelly Oil Company, summer 1968; has accepted an appointment as Systems Analyst for Skelly Oil Company, Tulsa, Oklahoma.

Organizations: Member of Triangle Fraternity, national, social, professional fraternity of engineers, architects, and scientists.